Shunra Analytics v8.6

Application Performance Engineering

Copyright Notice

Corporate Headquarters
1800 JFK Boulevard
Philadelphia PA 19103
USA
Tel: (215) 564 4046
Fax: (215) 564 4047
Sales: 1 877 474 8672
Support: 1 267 519 5137
info@shunra.com

European Office
73 Watling Street
London UK
EC4M 9BJ
Main: +44 (0)207 153 9835
Fax: +44 (0)207 785 6816
Sales: +44 (0)207 153 9835
Support: +44 (0)207 1539838
saleseurope@shunra.com

Israel Office
PO Box 7372
Hod Ha'sharon
Israel 45240
Tel: +972 9 764 3743
Fax: +972 9 764 3754
Sales: +972 9 763 4227
Support: +972 9 763 4227
info@shunra.com

Shunra Analytics- 8.6-(1.0)

Table of Contents

1

Shunra Analytics

Shunra Analytics assists in pinpointing factors that negatively impact an application's operation across a network. Shunra Analytics conducts an analysis based on packet list data, then displays the resulting data in informative reports that provide insight into an application's operation.

Analysis of HTTP, HTTPS and other protocols in waterfall diagrams provide a visual look into individual resource sizes and load times, enabling rapid analysis of transaction response times and the ability to quickly identify areas for optimization. In addition, the Shunra Analytics provides a method to automate the process of transaction analysis and grading.

This section describes:

- **Shunra Analytics Requirements**: (page 1-1) reviews the minimum host requirements for Shunra Analytics
- **Installing Software**: (page 1-2) provides step-by-step instructions on how to install Shunra Analytics
- **Licensing Shunra Analytics**: (page 1-3) provides instructions on how to license and activate Shunra Analytics

Shunra Analytics Requirements

The minimum requirements are as follows:

Processor	Dual core processor or higher
Memory	2 GB RAM or higher
Hard Disk	20 GB of free disk space

Operating System (English Version only)	Server 2003 SP2 Standard edition (32/64 bit) - Server 2008 SP2 Enterprise edition (32/64 bit) XP Professional SP3 (32 bit) XP Professional SP2 (64 bit) Windows 7 (32/64 bit)
Browsers	Internet Explorer 9.0 and higher Firefox 10.0 and higher Chrome 17.0 and higher
Additional Software	Microsoft Office 2007 and 2010: for export of reports to Word

Note: the following prerequisites will be installed during the installation if not already present:
- Silverlight 5.0 (x86)
- Microsoft .NET Framework 4.0 Full
- Java Runtime Environment 6.0 (x86)

Wireshark 1.6.2 to 1.6.8 can be downloaded during installation.

Installing Software

To install Shunra Analytics:

1. From the installation package, in the Shunra Analytics folder, run the Analytics_ setup.exe and follow the instructions in the wizard; reboot is required.
2. After installation, access the Shunra Analytics from the **Start** menu (**Start** > **Programs** > **Shunra > Analytics,** select **Shunra Analytics)**. Shunra Analytics opens in a browser window.

Silent Installation

Note: when the download and install of Wireshark (during setup) does not succeed, silent install will abort.

To silently install Shunra Analytics:

1. Copy the file Analytics_ setup.exe to a convenient location.

2. From the Start menu, choose Run; then type CMD.
3. In the Command window, navigate to the location of the file copied in step 1, and type:

   Analytics_ `setup.exe /s /v"PORT=<port number> [DATA_FOLDER=\"<path to data dir>\"] [INSTALLDIR=\"<path to install dir>\"]"`

   PORT: the port used by vCat is used by the Shunra Analytics if already installed; ENABLE_REMOTE adds the port to the firewall

   DATA_FOLDER: optional, default is the Common App Data folder ("C:\ProgramData" in Win 7).

   INSTALLDIR: optional

Silent Uninstallation

To silently uninstall Shunra Analytics:

1. Type:

   Analytics_ `setup.exe /s /removeonly /v"[PORT=<port number>]"`

   PORT option removes the port from the firewall

Upgrading Shunra Analytics

To upgrade to Shunra Analytics v8.6:

1. From the Shunra Installation folder, select the Analytics_setup.exe and follow the instructions in the wizard.

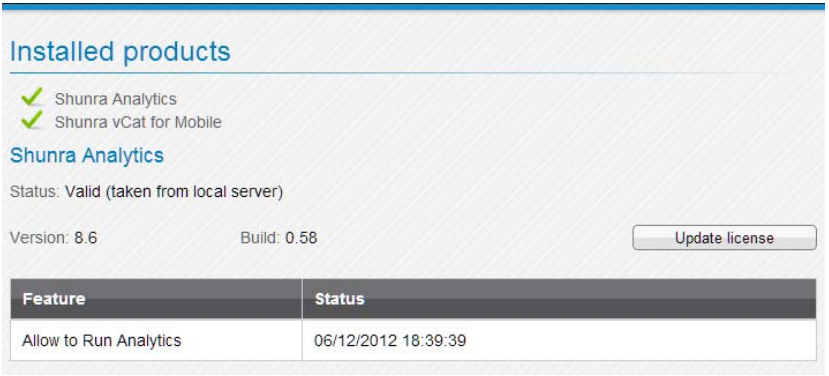
Licensing Shunra Analytics

Shunra Analytics comes with a 2 day trial license. To obtain an extended license, if you purchased Shunra products from HP Software, contact your HP Representative. Otherwise, follow these instructions:

To access the Shunra License Manager:

1. From the **Start** menu (**Start** > **Programs** > **Shunra)** select **Shunra License** (localhost:8182/Shunra/license).

The status of the license is displayed.



To obtain a new license:

Depending upon your license agreement, you can:

- Upload a license file (see **To upload a license file:**)
- Check out a license for a limited period from a local license server (see **To check out a license from a local license server (online):**)

To upload a license file:

1. In the **Update via**, select the **File** radio button.
2. Select **Download product key** and save the fingerprint file (.c2v) on the local machine.

3. Attach the product key file (.c2v) to an email message and sent to license@shunra.com

   Or

   Select **Send Shunra your customer details and attach product key** to attach the license file.
4. After the license file (.V2C) is received from Shunra, save the updated key file on the local machine.
5. Select the folder icon beside the File box and download the (.V2C) file.
6. Click **Update**. The updated license details are displayed in the Shunra License Manager main page.

To check out a license from a local license server (online):

Note: The offline method must be used when the license client cannot communicate with the licensing server using TCP port 1947; refer to the Shunra vCat License Server Installation and Configuration Guide.

---

1. From Start > Programs > Shunra, open the Shunra License Manager (localhost:8182/Shunra/license).
2. Select the **Update License** button.
3. Select **Local licensing server** radio button.
4. In the Local server address, select the machine on which the license server is installed. If the License Server does not appear in the list, but is reachable using TCP port 1947, type the license server hostname or IP in the Local server address box.
5. Enter the number of days required in the License duration box, and click Checkout license. The updated license details are displayed in the Shunra License Manager main page.

To check in a license to a local license server (online):

1. From Start > Programs > Shunra, open the Shunra License Manager (localhost:8182/Shunra/license).
2. Select the **Update License** button.
3. Select **Local licensing server** radio button.
4. Click **Return license**.

Log Files

Shunra products' log files are located in the <installation directory>\logs, by default in \Program Files\logs or \Program Files (x86)\Shunra\logs.

2

Analyzing Results

This section describes how to view and analyze the results generated by Shunra Analytics, including:

- **Creating Shunra Analytics Reports**: (page 2-2)
- **Secure Communication**: (page 2-22)

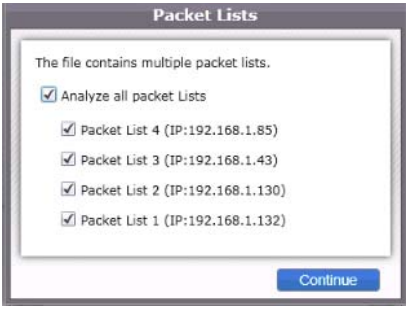
Creating Shunra Analytics Reports

Shunra Analytics reports consist of data captured using a network virtualization packet list data which is subsequently processed and displayed in intuitive reports. These reports are then examined to decipher problems that may exist.

The Analytics reports provide detailed data about the breakdown of each transaction. Statistics of each resource being uploaded or downloaded in a transaction are displayed in both tabular and graphic format. Precise performance data includes load times, component download analysis, response time breakdown and details of errors received. Performance optimization recommendations how to improve and optimize mobile and non-mobile transaction performance are provided.

To create Shunra Analytics report:

1. From the Start Menu, choose **Programs > Shunra > Analytics**.
2. Click Open File (to filter the port and define other settings, see **Settings**, page 2-3). The following file types are supported: *.shunra, *.ved, *.cap, *.pcap and *.enc.
3. When a file contains more than one Packet List, select some or all of the packet lists to be included in the analysis.



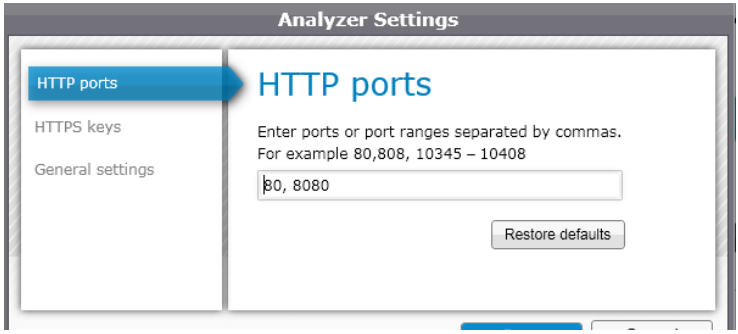
4. After the Shunra Analytics window opens, in the toolbar click **All Transactions** to select a display of all the transactions, or select a specific transaction. The main page displays all the transactions, and each 'square' shows one transaction.

Settings

To filter the analysis to a specific Port prior to analysis, click the Settings icon (wrench) on the Welcome page. The settings are also available from the main page.

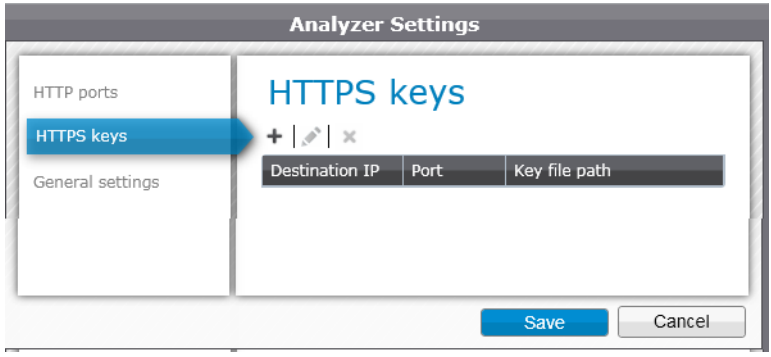
HTTP Port Settings

To modify the default port or port range, select the HTTP ports tab and add one or more ports or range, each separated by a comma.



HTTPS Keys

To enable analysis of secure data, enter the HTTPS key.



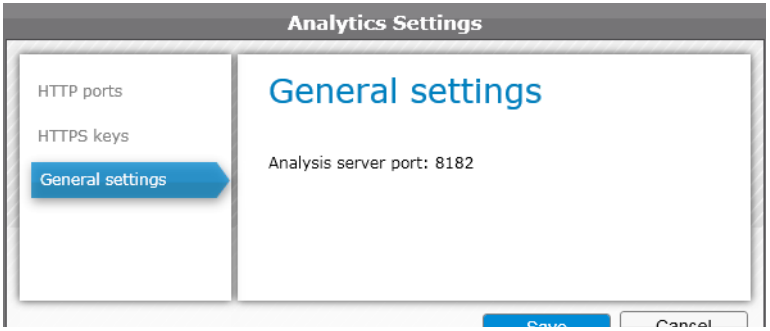
To add a HTTPS Key:

1 Click the "+" sign, and in the New HTTPS Key window, enter the required information.

2 To edit the information, click the pencil icon; to delete the Key click the "x".

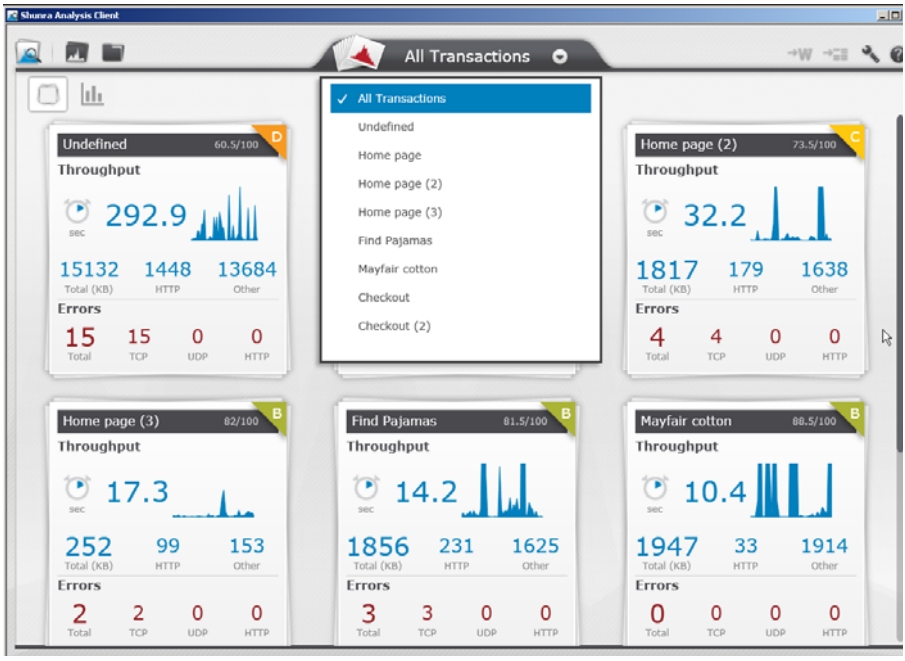
General Settings

Displays the Server port.



Overview

The Overview page is interactive, so clicking any transaction displays the details of that transaction in the relevant reports. Each Transaction section provides details of the throughput and errors, and shows a performance score with a letter and a percentage. The display is interactive, so that clicking any metric, such as "Total" displays the detailed report for that metric.



The Shunra Analytics reports Overview page displays detailed breakdown of each transaction in the following categories:

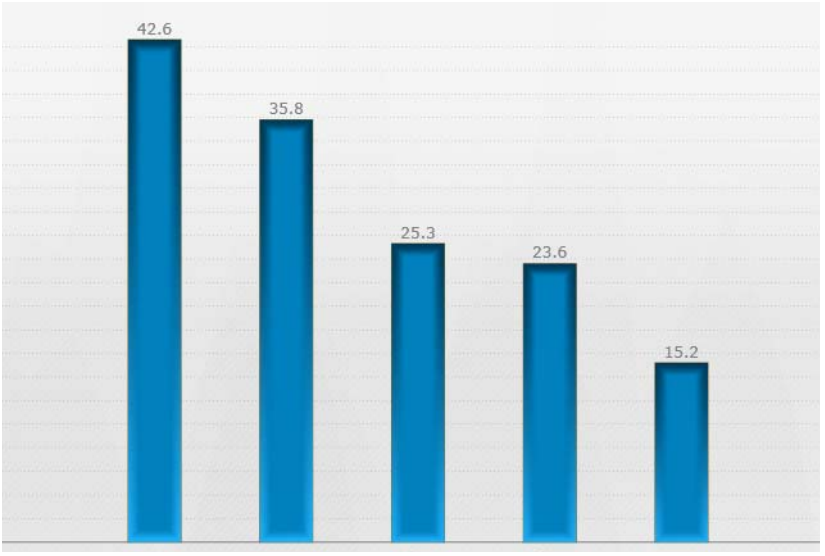
- **Response Time** (page 2-6)
- **Summaries** (page 2-6)
- **General Analysis** (page 2-8)
- **Endpoint Latencies** (page 2-12)
- **TCP/UDP Errors & Sessions** (page 2-13)
- **HTTP Analysis** (page 2-13)
- **HTTP Optimization** (page 2-18)
- **HTTP Resources and Responses** (page 2-19)

Below the list of reports (on the left side), the total TRT for the transaction is listed. If a description was added in the Shunra Transaction Manager it is also displayed.

In each of the report views, to return to the main page click the Home icon, or use the Back icon to return to the previous view, or the Forward icon to advance to the next view. A report that does not contain data displays an icon.

Response Time

In the All Transactions tab, click the  icon at the top left to display the Transaction Response Times of all transactions.

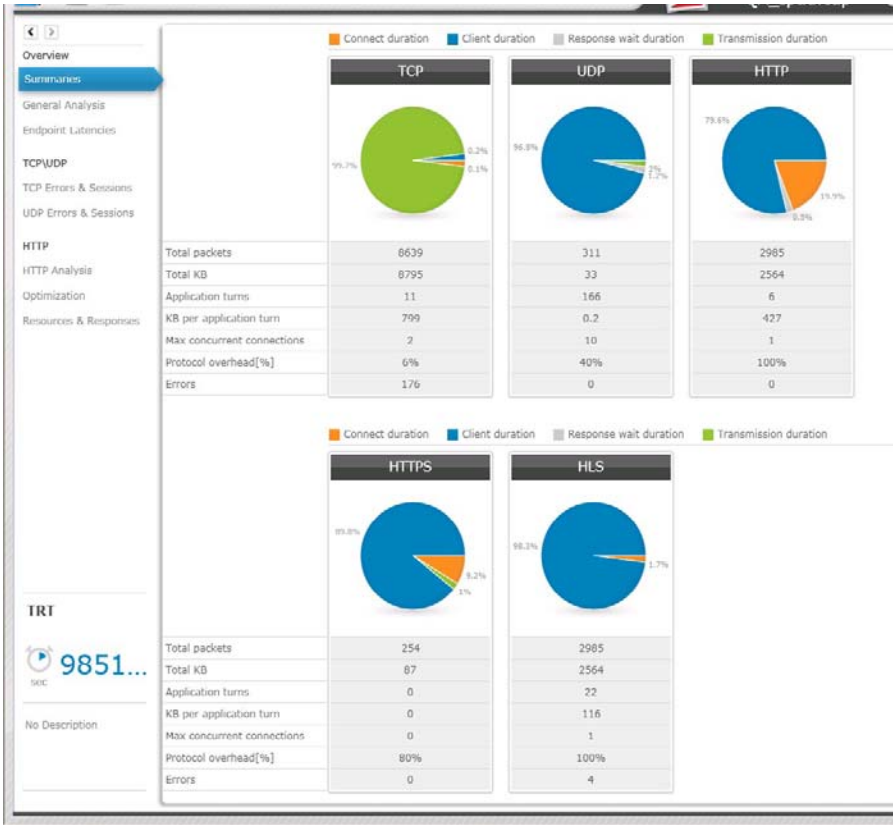


Click any bar to display the HTTP Analysis of that transaction.

Summaries

Displays the Client Network Server Breakdown of the transaction according to protocol, showing results in a pie chart and also additional details for the following protocols:

- TCP
- UCP
- HTTP
- HTTPS (secure communication)
- HLS (HTTP Live Streaming)



Client Network Server Breakdown

The legend for the chart is below the table. The values for the fields shown in each pie are:

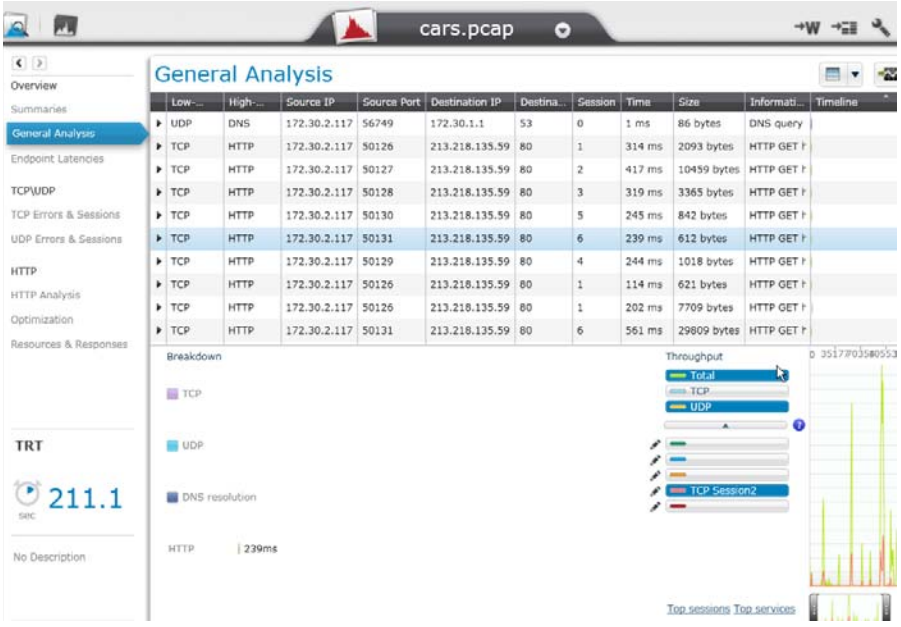
- **Connect duration**: portion of time in which the client connected to the server, such as the request in TCP or the triple handshake in SSL (the establishment of a secure channel)
- **Client duration**: portion of time that the client processes (does not include time waiting for a server response)
- **Response wait duration**: portion of time spent waiting for the server's response
- **Transmission duration**: portion of time data was downloaded or uploaded

Parameters

- **Total packets:** total number of packets associated with the protocol
- **Total KB**: total throughput associated with the protocol; includes headers, such as TCP, IP, etc.
- **Application turns:** the number of times a communication flow change occurs from the request to the response, per protocol
- **KB per application turn:** the average of the throughput per Application Turn, per protocol
- **Max concurrent connections:** the highest number of concurrent connections, per protocol
- **Protocol overhead %:** the percentage of the total throughput per protocol used by non-data elements such as headers
- **Errors:** the number of errors that occurred in the transaction

General Analysis

The General Analysis displays details of all subtransactions for all protocols.



Display Options

The following options are available from the toolbar of the General Analysis and HTTP Analysis reports. The first two options are also available in the right-click menu in the table:

- Highlight options: click  to select from the various options that highlight the resources according to the same Source IP, etc.
- Display in graph options: click  to select to display a session, service, etc. in the graph
- Filter options: click  to select parameters that limit the display of the subtransactions to the selected criteria

Subtransaction Parameters

To adjust the display of the subtranscations:

- Sort the rows according to ascending or descending order in each column (available in all tables)
- Expand or decrease the area by dragging the borders of the area (available in the General and HTTP Analysis).

The following data is provided in table format. Each row represents a sub-transaction:

- **Low-level Protocol**: includes TCP and UDP
- **High-level Protocol**: includes HTTP and HTTPS, DNS, etc.
- **Source IP**
- **Source Port**
- **Destination IP**
- **Destination Port**
- **Session**: number of the TCP or UDP session in which the resource was uploaded or downloaded; color coding scheme indicates the tasks occurring in the download, such as DNS resolution, TCP setup, etc.
- **Time**: response time of the request in milliseconds
- **Size**: the number of bytes used in the request/response
- **Information**: protocol specific information (when available)
- **Timeline**: the position of the Resource within the Transaction sequence

Request/Response Details

To display the Request/Response details, expand or collapse a row with the ▸ icon in the left column. The Client and Server portion of each Request/Response are displayed in bytes.

Breakdown

The breakdown of the share of the traffic for each protocol is displayed in the graph at the lower left of the window, including TCP, UDP, DNS Resolution and HTTP.

Throughput Graph

Note: The breakdown and throughput area can be collapsed and expanded by clicking the double arrow below the main table.

Throughput

The Throughput edit options are used to modify the display in the graph.

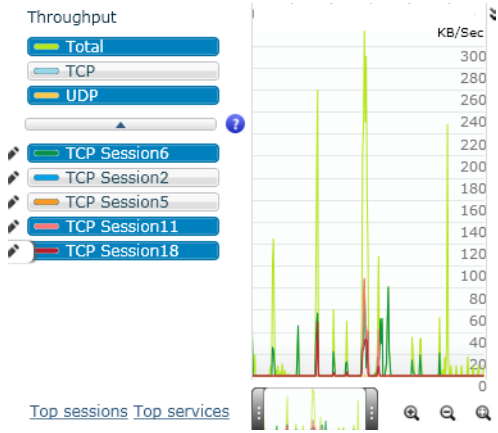
Use the zoom bar to focus on a specific period within the transaction.

- **Top Sessions**: displays up to five Sessions with the most traffic
- **Top Services**: displays up to five Services with the most traffic

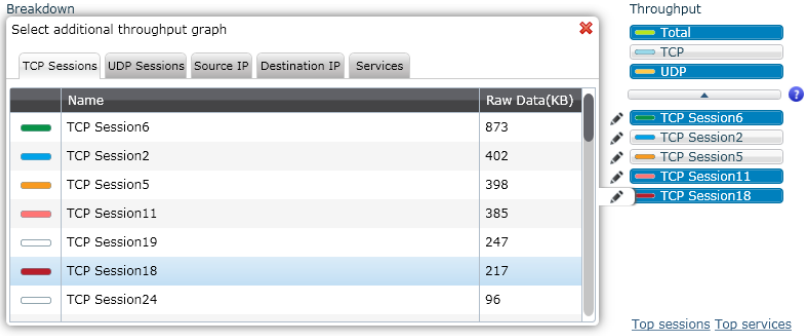
In each tab of the Throughput area, click a bar to include that item, such as a specific session in the graph. The Raw data displays the throughput in KB/sec.

To add/delete components in the Throughput graph display:

1. Click one of the tabs, for example "TCP Session 18"; the graph displays the throughput for TCP Session 18 in maroon.



2 To select other sessions, services, etc., click the pencil icon for each session or hosts that is to be displayed.



3 The following tabs are available and each displays the traffic (in KB/Sec) for each of the following values:

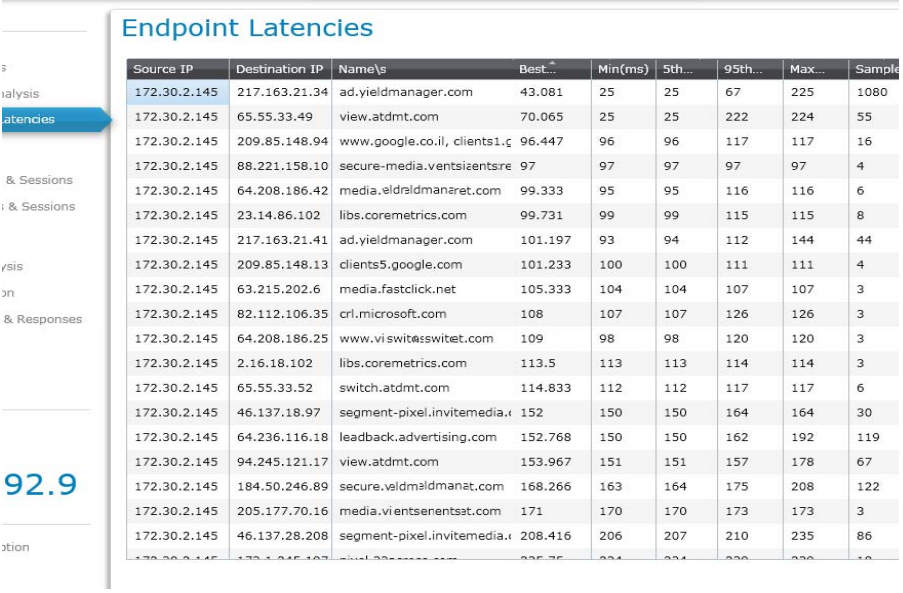
- TCP Sessions
- UDP Session
- Source IP
- Destination IP
- Services (IP Address and Port)

4 Select the required item, for example "TCP session 16" or a Service Name. The metric is displayed in the graph with the selected color.

Endpoint Latencies

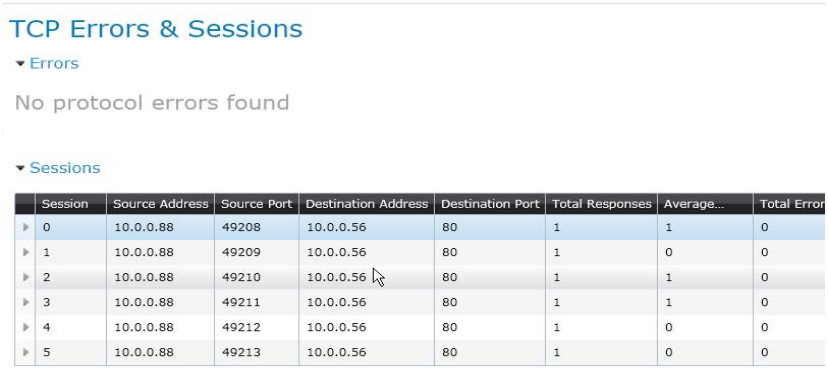
The Endpoint Latencies report displays details of the latency observed at the client and server endpoints.

- **Source IP**
- **Destination IP**
- **Names:** The name of the server
- **Best estimate (ms)**: Best estimate of the latency between the client and server, as deduced from the TCP connections between the two, taking into consideration any additional latency found in the packet capture due to bandwidth constraints (thus, it's possible for this value to be lower than the Min value)
- **Min (ms):** the minimum value observed in the packet capture
- **95th percentile (ms):** the maximum value observed in the packet capture, excluding outermost conditions
- **Max (ms):** the maximum value observed in the packet capture
- **Samples:** the number of packets used in the calculation of the latency



TCP/UDP Errors & Sessions

Select **TCP errors & sessions** to display the details of transaction that used the TCP protocol. Select **UDP errors & sessions** to display the details of transaction that used the UDP protocol.



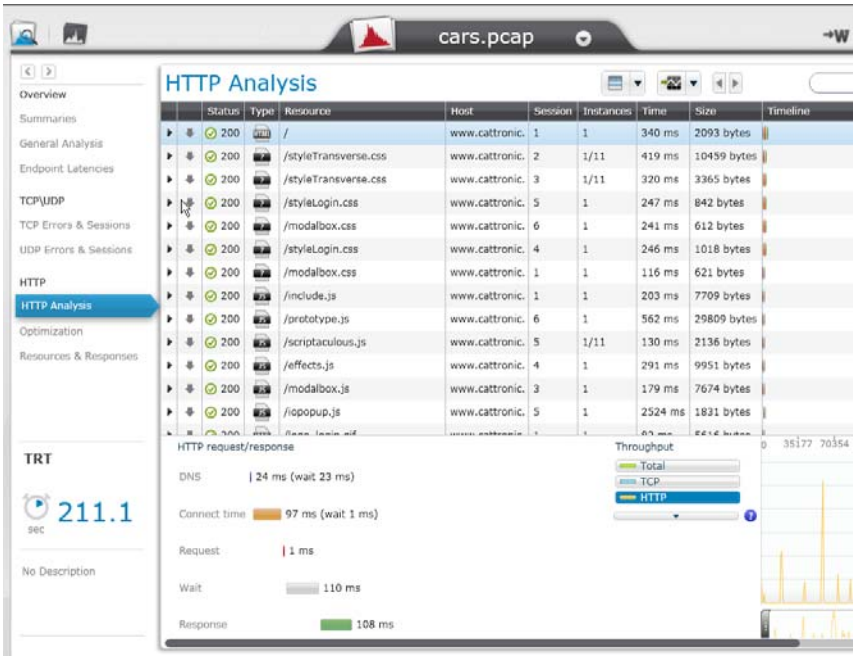
Session	Source Address	Source Port	Destination Address	Destination Port	Total Responses	Average...	Total Error
0	10.0.0.88	49208	10.0.0.56	80	1	1	0
1	10.0.0.88	49209	10.0.0.56	80	1	0	0
2	10.0.0.88	49210	10.0.0.56	80	1	1	0
3	10.0.0.88	49211	10.0.0.56	80	1	1	0
4	10.0.0.88	49212	10.0.0.56	80	1	0	0
5	10.0.0.88	49213	10.0.0.56	80	1	0	0

Session details for TCP and UDP are displayed separately, and each includes:

- **Session**
- **Source Address**
- **Source Port**
- **Destination Address**
- **Destination Port**
- **Total Responses**
- **Average Response Time**
- **Total Errors**
- **Error Types**

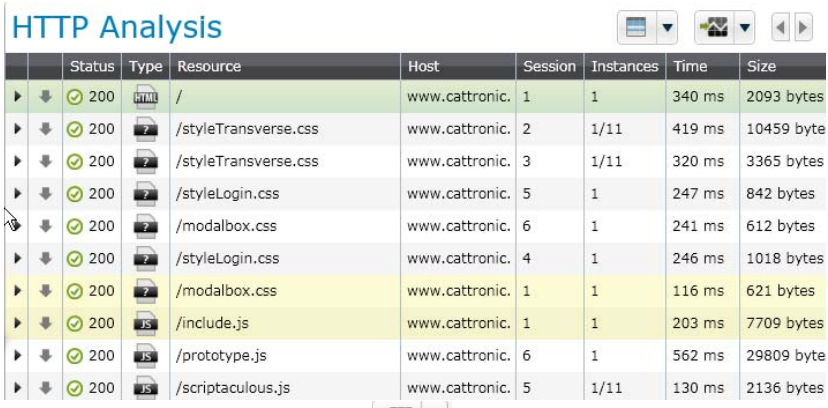
HTTP Analysis

The details of the transaction are displayed in both table and graph format.



The following options are available in the toolbar of the General Analysis and HTTP Analysis reports. The first two options are also available in the right-click menu in the table:

- Highlight options: click  to select from the various options that highlight the resources according to the same Source IP, etc. This example shows all the Session's participants.

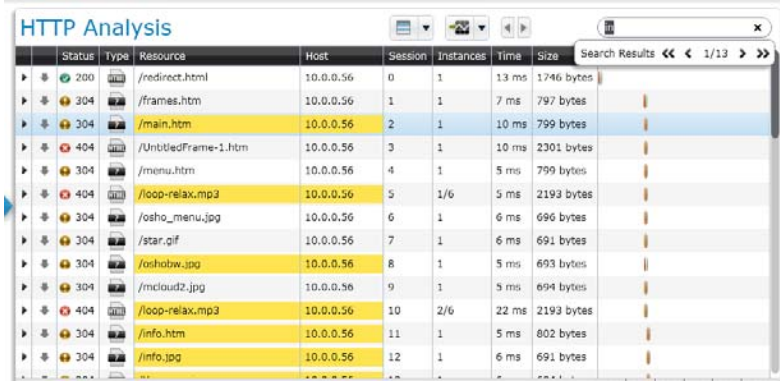


		Status	Type	Resource	Host	Session	Instances	Time	Size
▸	⬇	200	HTML	/	www.cattronic.	1	1	340 ms	2093 bytes
▸	⬇	200	?	/styleTransverse.css	www.cattronic.	2	1/11	419 ms	10459 byte
▸	⬇	200	?	/styleTransverse.css	www.cattronic.	3	1/11	320 ms	3365 bytes
▸	⬇	200	?	/styleLogin.css	www.cattronic.	5	1	247 ms	842 bytes
▸	⬇	200	?	/modalbox.css	www.cattronic.	6	1	241 ms	612 bytes
▸	⬇	200	?	/styleLogin.css	www.cattronic.	4	1	246 ms	1018 bytes
▸	⬇	200	?	/modalbox.css	www.cattronic.	1	1	116 ms	621 bytes
▸	⬇	200	JS	/include.js	www.cattronic.	1	1	203 ms	7709 bytes
▸	⬇	200	JS	/prototype.js	www.cattronic.	6	1	562 ms	29809 byte
▸	⬇	200	JS	/scriptaculous.js	www.cattronic.	5	1/11	130 ms	2136 bytes

✦ Display in graph options: click  to select to display a session, service, etc. in the graph

To search for similar resources according to URL:

1 In the toolbar, enter the required string in the Search area, in this example "in".



		Status	Type	Resource	Host	Session	Instances	Time	Size
▸	⬇	200	HTML	/redirect.html	10.0.0.56	0	1	13 ms	1746 bytes
▸	⬇	304	?	/frames.htm	10.0.0.56	1	1	7 ms	797 bytes
▸	⬇	304	?	/main.htm	10.0.0.56	2	1	10 ms	799 bytes
▸	⬇	404	HTML	/UntitledFrame-1.htm	10.0.0.56	3	1	10 ms	2301 bytes
▸	⬇	304	?	/menu.htm	10.0.0.56	4	1	5 ms	799 bytes
▸	⬇	404	HTML	/loop-relax.mp3	10.0.0.56	5	1/6	5 ms	2193 bytes
▸	⬇	304	?	/osho_menu.jpg	10.0.0.56	6	1	6 ms	696 bytes
▸	⬇	304	?	/star.gif	10.0.0.56	7	1	6 ms	691 bytes
▸	⬇	304	?	/oshobw.jpg	10.0.0.56	8	1	5 ms	693 bytes
▸	⬇	304	?	/mcloud2.jpg	10.0.0.56	9	1	5 ms	694 bytes
▸	⬇	404	HTML	/loop-relax.mp3	10.0.0.56	10	2/6	22 ms	2193 bytes
▸	⬇	304	?	/info.htm	10.0.0.56	11	1	5 ms	802 bytes
▸	⬇	304	?	/info.jpg	10.0.0.56	12	1	6 ms	691 bytes

All matching resources are highlighted, in this case strings that contain "in". Use the arrows below the Search area to navigate between the results.

2 Clear the search results by closing the Search area (clicking the "X").

Subtransaction Details

To adjust the display of the request/responses:

- Sort the rows according to ascending or descending order in each column
- Expand or decrease the area by dragging the borders of the area.

The following columns are displayed in the report:

- **Expand/Collapse** icon : show or hide details of each Request
- **Up/down arrows:** the up arrow indicates a POST or PUT; down arrow indicates a GET; a head indicates an icon header; N/A indicates HTTPS; for all other types a star is shown
- **Status**: the HTTP status, such as 404 (page not found) or 200 (OK)
- **Type**: the icon indicates the type of file requested, for example a graphic file.
- **Resource**: displays the path that was accessed by the subtransaction
- **Host**: the host (domain, server, etc.) from which the resource is uploaded or downloaded
- **Session**: number of the TCP session in which the resource was uploaded or downloaded
- **Instances**: number of times a resource with the same name appears in the transaction
- **Time**: response time of the request in milliseconds
- **Size**: the number of bytes used in the request/response
- **Timeline**: the position of the Resource within the Transaction sequence

The following data is displayed per Transaction (at the left):

- **TRT:** the time it takes to transaction to complete (between the first packet of the request and the last packet of the transaction)
- **Description:** displays the description provided for the transaction in the Shunra Transaction Manager.

To customize the display of the graph:

- The graph at the bottom-right displays the throughput. The 'handles' can be used to focus on a specific period within the Request/Response.
- Select HTTP Throughput to display the HTTP data only, or Total Throughput to display all the data in the Request/Response.
- When the mouse moves over this area, a line moves with the mouse to connect the Request/Response area and the graph view, and indicates the time (in seconds) when this occurred during the capture. Use the zoom

icons to zoom in, zoom out and to zoom out or to select no zoom
 .

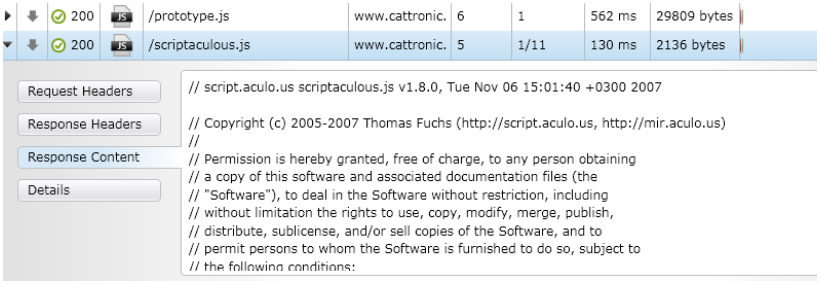
HTTP Parameters

When a resource is highlighted, the area in the below the table displays the following breakdown:

- **DNS Resolution**: includes the wait time from the resolve time to the start of the connection (the time between the DNS query and the first SYN to the server whose name was queried)
- **Connect Time**: the TCP Setup time and any wait time between the initialization of the connection and sending of the first request data packet
- **TLS Time:** SSL/TLS secure channel establishment
- **Request:** time required for the client to send the request to the server
- **Wait:** the time (in ms) between the last packet of the request and the first packet of the response
- **Response:** the time (in ms) between the first packet and the last packet of the response
- **Encrypted Data Transmission:** the duration of an encrypted HTTPS session

To display the details of a Request/Response:

1. Double-click the Request/Response name, or click the Expand/Collapse icon; the following columns are displayed:
   - **Request Headers**: the syntax of the request header
   - **Response Headers**: the syntax of the response header
   - **Request Content/Response Content**: displays the image, HTML data, etc. (non-printable characters may be replaced by a " . " Content in certain formats such as PDF, Audio, Video, Flash, and Fonts are not displayed.
   - **Details**: throughput of the request/response; for a HTTP POST the full URL is also displayed.



Note: When the Resource is selected (highlighted), details of the Resource are displayed below.

HTTP Optimization

Shunra provides a number of Best Practice recommendations, based on data obtained from external sources, in addition to the knowledge obtained within Shunra from current application testing methodology.

Each transaction is given a score comparing the transaction to the best practice. The Total Score (out of 100) is the summary of the individual scores, based on the level of compliance to website programming rules. Prioritization emphasizes which transactions most affect the results. Each recommendation is weighted, based on the potential performance improvement to be obtained if the recommendation is performed, and is displayed in a negative numeric point value, such as -4 Points. In addition, a value from A - F is provided; F indicates that a significant improvement would be obtained in reducing transaction response time by implementing the recommendation, whereas A indicates that minimal benefit will be obtained.

To display the list of transactions with the relevant recommendations:

1. Select **Optimization** and select the rules to be included in the report; by default all are selected. The report can be displayed more than once with different rules.
2. Select **Desktop** or **Mobile**; you can also sort the results according to the **Priority** (according to the number of points given to each result) or by **Name** (alphabetical order).
3. To display additional details, click a recommendation. For example, for the recommendation "Don't download the same data twice", the details show the number of times the file appeared. Each can be clicked and viewed in the HTTP Analysis.

Note: The score and recommendations may be different for desktop and mobile results, due to differences in browser functionality.

To filter the list of rules for both mobile and desktop:

1. Click the Edit (pencil) icon at the top right.
2. Select or deselect a rule, then click **Save**. When only some of the rules are selected, an "information icon" appears beside the Edit icon .

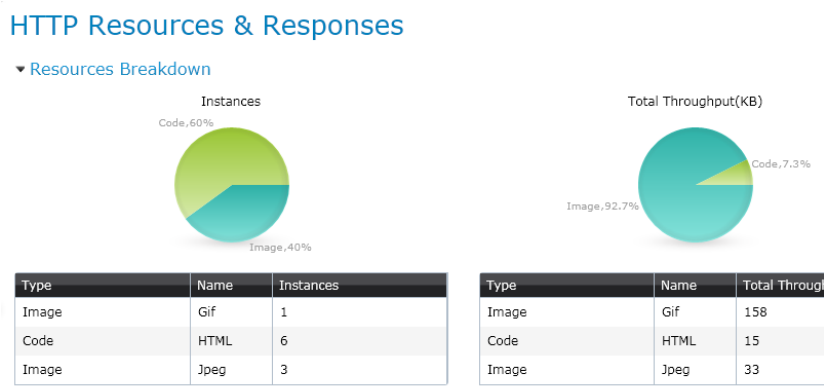
HTTP Resources and Responses

The Resources Breakdown displays the Instances and Total Throughput for each type of resource that was present in the transaction.

Resources Breakdown

The Instances pie chart shows the number of times that each type of resource appears in the transaction. The chart is divided according to type of resource, so that .jpeg images are one category, and css files are another category.

The Total Throughput chart shows the breakdown according to the same categories as the Instances chart, but the size is calculated according to the Total Throughput in KB.



Type	Name	Instances
Image	Gif	1
Code	HTML	6
Image	Jpeg	3

Type	Name	Total Throug
Image	Gif	158
Code	HTML	15
Image	Jpeg	33

HTTP Errors

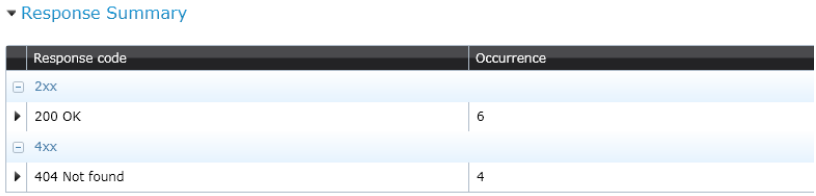
The table displays each error according to the subtransaction for which it occurred. The Errors by Class graph displays the totals according to the class, such as 4xx for HTTP client errors, and 5xx for server errors.

HLS Errors

The table displays a summary of errors that occurred during video streaming.

Response Summary

Displays the number of occurrences of each HTTP response code.



Exporting Shunra Analytics Results

Data of the selected view and recommendations can be exported to a .csv file and in MS Word format. The export to .csv includes the throughput, the network conditions displayed in the waterfall graph, and recommendations such as the rules, grade achieved and the violations for each rule.

To export results in .csv format:

1. Click the  icon in the toolbar.

   Note: When using a zoom view, the export will use the selected portion of the transaction, and not export the entire transaction.

2. In the Export settings dialog, select all or some of the reports, and any or all of the options, then click **Export**.

When exporting in MS Word format, the HTTP Report and Recommendations can be exported.

Note: When using a zoom view, the export will use the selected portion of the transaction, and not export the entire transaction. The Shunra Analytics template file must be installed in its original location, usually in the Documents folder.

To export results in MS Word format:

1. Click the  icon in the toolbar.

Note: When using a zoom view, the export will use the selected portion of the transaction, and not export the entire transaction. Results of searches that are highlighted are also highlighted in the exported reports.

Secure Communication

Secure HTTPS communication can be viewed in the Shunra Analytics Report, including the Host Name, TCP Session Establishment and SSL Session Establishment data.

Note: When the private key is not known or unavailable, a debugging proxy such as Fiddler, Charles or Burp can be utilized.

The private key must be a text file in the Privacy Enhanced Mail (PEM) format, as in this example:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC20yQBpiE1atCfIuiOqa9fnPyHrCsYCoCJ/hKlS8z6IOcmBsNq
kZckzlMqcQP7i9o3SGVBXv1rAsGN3SnkqNUD4PFukUHQrjPpc0KPX9KdDCcLFu5f
Bxq+7/7hDzQA+rKWqixO3WmBVcKQm+3WvpF5M+jJuIsRY8O6rOxb+FrpvwIDAQAB
AoGBAKWpszHPUL4vmPTqU+KZ/bDc9rMVnkL9mTXxRQIFKqroT6vUaxTQ8i1GzHNj
zyELu+NmNWJD6cwixjJ/Wmu5VLLT9syvBnAuLNDwn02/TrOurxdk/FeEuPEF6exo
GKKDAAzDCYvEBDOEbYBx/3s+v0n3XLeW6dKX76JZkv97LokZAkEA2kO1u+WDAGcB
s5pq3I2RlOELeao4d/1dY7x4Efap3HJNWMF0byZEwPyC5yKkEZQDKt3n549nTPxM
wDO9geHjqwJBANZu4lxS/+4WkqwN5yzh8VCKmMUc0PQvXw+niXrjSucc5k5VdHw4
qOobXalp1PyrafHj6YV8PfX9XHpPRAzs/j0CQBq8iJqHttBpzc+ObuCthtPR7XHt
5BTuuG3rPFoW+R/D8K2apQSoj2uEgxSFLcvpcaninPHEo0bO8SfqLqCmZxkCQADb
dKA13LwQ7wktDQ2K4bIWu8Gd+d/gCJtBajVJj1UZMnqBsPOGLnaxIVC6EZXpAYVs
CdT0yDKhjqsWggkjMWkCQHZvP1E28M51k1pLsQx43nq7zbueKZWkDg/biA3yOaLb
FJ9TSJeufAXAmG/US+zCfgLuzrSuJwHiCMnhRrBOm+Y=
-----END RSA PRIVATE KEY-----
```

If it is not in this format, use Open SSL to change the format of the private key. Determine the operating system of the Server. On Microsoft® Windows, keys are often stored in PKCS7/DER format (locally) or in .NET format (from any directory server). To convert use these commands:

```
# for PKCS7/DER keys (as held on disk)
openssl pkcs8 -nocrypt -in derfile.key -inform
DER -out key.pem -outform PEM
# for NET keys (from the directory server)
openssl pkcs8 -nocrypt -in file.ick -inform NET -
out key.pem -outform PEM
```

On Mac OSX, Solaris, and other systems the file format used is often PKCS#12. To convert use this command:

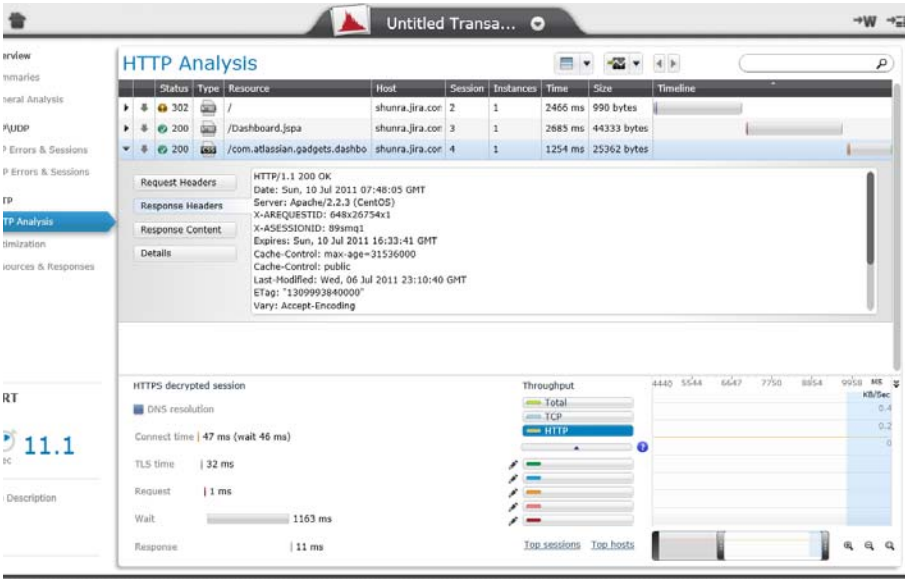
```
openssl pkcs12 -nodes -in file.p12 -out key.pem -
nocerts -nodes
```

On Linux use these commands:

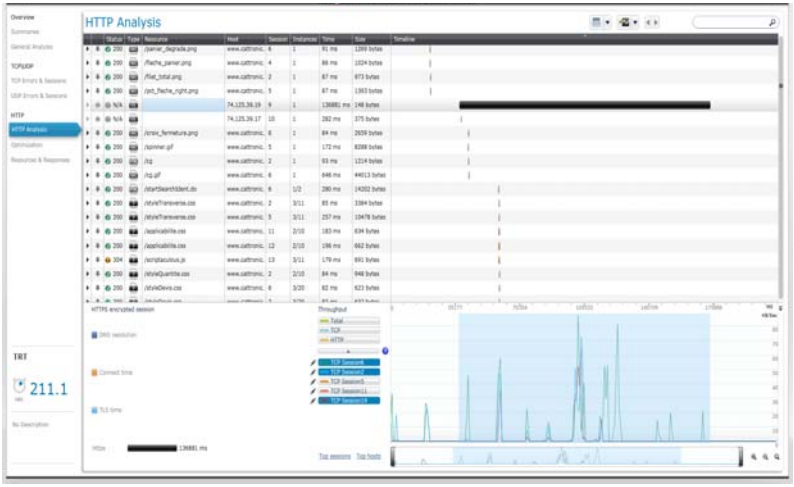
```
openssl x509 -nocrypt -in foo.der -informat DER -
out key.pem -outformat PEM
openssl x509 -nocrypt -in foo.net -informat NET -
out key.pem -outformat PEM
```

Note: To analyze secure communication when the server's private key is known, refer to the SSL options in **Adding Applications and Ports** on page 2-7.

This is an example of the detailed view when HTTPS traffic is decrypted.



When the traffic is not decrypted, the transaction details are not visible, as in this example.


HTTP Analysis
TRT
211.1

3

Shunra Analytics API

Shunra's API uses Representation State Transfer (REST) web services architecture. The analysis API requests all have a same URL structure, the prefix is:
`[base address]/shunra/api/analysis`

Note: A code sample "analyzer.py" in Python is available in the installation folder. Updates can be found in https://gist.github.com/2773832
It can be used to access the API. Segments of the code are also provided in this document.

The following methods are exposed:

1. **AnalysisEngines**
2. **Extract Packet Lists**
3. **AnalysisRequest**
4. **AnalysisSummary**
5. **AnalysisArtifact**

AnalysisEngines

Provides a JSON contents list of the installed analysis engines.

GET

[base address]/shunra/api/analysis/engines
`http://localhost:8182/shunra/api/analysis/engines`

Response

The response includes the id and names of all the analysis engines.

{
"supportedAnalysisEngines":[{"name":"harExport","id":"har
Export"},{"name":"networkmeasurements","id":"networkmeasu
rements"},{"name":"generalWaterfall","id":"generalWaterfa
ll"},{"name":"http","id":"http"},{"name":"iostats","id":"
iostats"},{"name":"metrics","id":"metrics"},{"name":"best
practices","id":"best practices"}]}
}

Returns

- 200 "OK"
- 404 "Not Found"
- 500 "Internal Server Error"

Code sample

```
def get_engine_id(engine_name):
    """
    Returns the analysis engine id, given its name.
    This can also be used as a sort of a sanity test for
    the analysis api.

    >>> get_engine_id('best practices')
    u'best practices'
    """
    resp = get('/shunra/api/analysis/engines')
    engines = dict([(entry['name'], entry['id']) for
    entry in resp['supportedAnalysisEngines']])
    return engines[engine_name]
```

Extract Packet Lists

Provides a JSON contents of packet list names, IDs, endpoints and .pcap and .ved file unique IDs.

PUT

[base address]/shunra/api/analysis/packetlistmetadata
http://localhost:8182/shunra/api/analysis/packetlistmetadata

Body

The JSON defines the analyzed emulation result (.ved or .pcap file) ID. It is a file system path for the Shunra Analytics.:

```
{
"id":"C:\\tmp\\Sample.ved"
}
```


The response includes the ID of the analyzed run result and the packet list metadata (names, IDs, endpoints):


```
{
"packetLists":[{"endpoints":[{"name":"Tokyo
Office","id":"6d0652db88c349de9382a54dc350349f"}],"name":
"Packet List
3","id":"c6064d9bf25d405382e374795fef35fe"},{"endpoints":
[{"name":"London
Office","id":"de358779547c4eea8caeef62bfbbb493"}],"name":
"Packet List
2","id":"59220e1cb4d248eba3b89a695918be91"},{"endpoints":
[{"name":"NY
Office","id":"8c95498f7bb04c7598dde1d5e609082a"}],"name":
"Packet List 1",
"id":"620984c9a31b4ef694a1ac47d61b6a7e"}],"runResultId":"
b80de7f5ffa97428b2324c8b3a9d469b"
}
```


Returns

- 200 "OK"
- 404 "Not Found"
- 500 "Internal Server Error"

Code sample

```
def get_packetlists(inputfilepath):
    """
    Returns a dictionary of the available packet lists in
    the given file.
    The dictionary keys are the packet lists names, and
    the dictionary values are the packet lists ids.

    >>> packetlists =
    get_packetlists(os.path.join(SAMPLE_FOLDER,
    'Sample.ved'))
    >>> len(packetlists)
    3
    >>> 'Packet List 1' in packetlists
    True
    """
    resp = put('/shunra/api/analysis/packetlistmetadata',
     {'id':inputfilepath})
```

```
    return dict([(entry['name'], entry['id']) for entry in
     resp['packetLists']])

def get_run_result_id(inputfilepath):
    resp = put('/shunra/api/analysis/packetlistmetadata',
     {'id':inputfilepath})
     return resp['runResultId']
```

AnalysisRequest

Represented by JSON; provides contents of current status of analysis process per packet list, per transaction and per analysis engine.

The response is a dictionary with the following entries:

- **transactionAnalysisStatus** - a list of transactions as described below
- **reportId** - an identifier for the analysis process
- **name** - name of the analyzed packet list
- **id** - id of the analyzed packet list

The list of transactions contains an entry for each transaction assoicated with the packet list.

Each entry is a dictionary, containing the transaction id (id), name (name) and analysis status (analysisStatusPerEngine).

Analysis status is a dictionary whose keys are the analysis engine, and the values are their status as specified in the API documentation.

PUT

[base address]/shunra/api/analysis/request/{plid}
http://localhost:8182/shunra/api/analysis/request/620984c9a31b4ef694a1ac47d61b6a7e

Where "plid" is a packet list unique ID that has been returned by the Extract Packet List request.

Body

Contains the analysis parameters such as ports, SSL Encryption Key, and the analyzed emulation result (.ved or .pcap file) ID. and the file system path, because the file is not persisted (retained?) by the system. The body is in JSON format:

```
{
```

```
"ports":"80, 8080",
"sslEncryptionKey":
"172.30.2.31,443,http,C:\\keys\\secret.key",
"runResultHandle":"C:\\tmp\\Sample.ved"
}
```

The response includes a current analysis status per transaction, per installed analysis engines and the generated analysis report id identifying the analysis parameters:

```
{
"transactionAnalysisStatus":[{"analysisStatusPerEngine":{
"networkmeasurements":"Started","harExport":"Started","ge
neralWaterfall":"Started","http":"Started","iostats":"Sta
rted","metrics":"Started","best
practices":"Started"},"name":"Undefined","id":"ccb8713e52
2241c9a691c4ed1ce72d27"}],"reportId":"-
561678026","name":"Packet List
1","id":"620984c9a31b4ef694a1ac47d61b6a7e"
}
```

Where possible analysis statuses are:

```
public enum WorkStatus {
  // a job still has not been started, not proceeded,
analyzed, etc
  Idle(0),
  // a job (for example emulation or analysis) started
  Started(1),
  // a job (for example emulation or analysis) finished
  Finished(2),
  // a job (for example analysis) failed
  Failed(3);
}
```

Note: The heuristic for analysis process completeness is that all the items have either a Finished or Failed status; otherwise some items in the analysis jobs pool have not completed yet.

The client side should continue to process analysis requests until the analysis process has completed.

Returns

- 200 "OK"
- 404 "Not Found"
- 500 "Internal Server Error"

Code Sample

```
def analyze(inputfilepath, packetlist_id, settings={}):
     """
     calls analysis on a given file (use settings to pass
     special analysis parameters such as port numbers and
     ssl keys)

     packetlist_id should be the id return by
     get_packetlists for a specific packet list.


     The response is a dictionary with the following
     entries:
          * transactionAnalysisStatus - a list of
          transactions as described below
          * reportId - an identifier for the analysis
          process
          * name - name of the analyzed packet list
          * id - id of the analyzed packet list

     The list of transactions contains an entry for each
     transaction assoicated with the packet list.
     Each entry is a dictionary, containing the
     transaction id (id), name (name) and analysis status
     (analysisStatusPerEngine).
     Analysis status is a dictionary whose keys are the
     analysis engine, and the values are their status as
     specified in the API documentation.

     >>> inputfilepath = os.path.join(SAMPLE_FOLDER,
     'Sample.ved')
     >>> packetlists = get_packetlists(inputfilepath)
     >>> packetlist_id = packetlists['Packet List 1']
     >>> resp = start_analysis(inputfilepath,
     packetlist_id)['transactionAnalysisStatus']
     >>> len(resp) # only one tranasction is associated
     with this packet list
     1
     >> resp[0]['name']
     u'Undefined'
     >>> resp[0]['analysisStatusPerEngine']['http'] in
     ['Idle', 'Started', 'Finished', 'Failed']
     True

     """
     params = dict(settings)
     params['runResultHandle'] = inputfilepath
```

```
        resp = put('/shunra/api/analysis/request/
        '+packetlist_id, params)
        return resp
    def get_report_id(inputfilepath, packetlist_id,
        settings={}):
        return analyze(inputfilepath, packetlist_id,
        settings)['reportId']

    def get_transactions(inputfilepath, packetlist_id,
        settings={}):
        """
        Gets all the transactions associated with a given
        packetlist.
        The result is a list of pairs, the first element of
        each pair is the transaction id, and the second is
        the transaction's name

        >>> inputfilepath = os.path.join(SAMPLE_FOLDER,
        'Sample.ved')
        >>> packetlists = get_packetlists(inputfilepath)
        >>> packetlist_id = packetlists['Packet List 1']
        >>> result = get_transactions(inputfilepath,
        packetlist_id)
        >>> len(result) # only one transaction is associated
        with this packet list
        1
        >>> result[0][1]
        u'Undefined'
        """
        return [(transaction['id'], transaction['name']) for
        transaction in analyze(inputfilepath, packetlist_id,
        settings)['transactionAnalysisStatus']]


    def start_analysis(inputfilepath, packetlist_id,
        settings={}):
        """
        Starts analysis on a given file.

        The response is a list, with an entry for each
        transaction assoicated with the packet list.
        Each entry is a dictionary, containing the
        transaction id (id), name (name) and analysis status
        (analysisStatusPerEngine).
        Analysis status is a dictionary whose keys are the
        analysis engine, and the values are their status as
        specified in the API documentation.
        """
```

```
        return analyze(inputfilepath, packetlist_id,
        settings)

def is_analysis_done(inputfilepath, packetlist_id,
settings={}):
        """
        Returns True if all the transactions associate with
        the given packet list were analyzed and their reports
        are ready to be fetched.

        """
        resp = analyze(inputfilepath, packetlist_id,
        settings)['transactionAnalysisStatus']
            for transaction in resp:
                for engine_status in
        transaction['analysisStatusPerEngine'].values():
               if engine_status in ['Idle','Started']:
                   return False
    return True
```

AnalysisSummary

Represented by JSON; provides the contents of analysis summary per packet list, per transaction, and per analysis engine.

GET

[base address]/shunra/api/analysis/summary/{runresulthandle}/{plid}/{reportId}/{engineId}

```
http://localhost:8182/shunra/api/analysis/summary/
b80de7f5ffa97428b2324c8b3a9d469b /
620984c9a31b4ef694a1ac47d61b6a7e/-561678026/
best%20practice
```

GET

[base address]/shunra/api/analysis/summary/{runresulthandle}/{plid}/{reportId}/{trId}/{engineId}

```
http://localhost:8182/shunra/api/analysis/summary/
b80de7f5ffa97428b2324c8b3a9d469b /
620984c9a31b4ef694a1ac47d61b6a7e/-561678026/
best%20practices
```

The first call returns the requested analysis report for all transactions in packet list. The second returns it for the specified transaction only.

The following report types are currently supported:

- **http**: HTTP Analysis
- **best practices**: Optimization report
- **iostats:** Throughput report
- **general/waterfall**: General Analysis
- **metrics**: the protocols' summary and metrics report
- **networkmeasurements**: Endpoints Latencies report
- **harExport**: a report containing the HTTP subtransaction in HAR format (experimental)

Response

GET

'/shunra/api/analysis/summary/%s/%s/%s/%s/%s'%(run_result_handle, packetlist_id, report_id, transaction_id, engine_id)

Returns

resp['successfullTransactionAnalysis'][0]['result'] ???

- 200 "OK"
- 404 "Not Found"
- 500 "Internal Server Error"

Code Sample

Code Sample

```
def get_analysis_report(run_result_handle, packetlist_id,
    report_id, transaction_id, engine_id):
    """
    Get the result of running one of the analysis engines
    on a given packet list
    """

    resp = get('/shunra/api/analysis/summary/%s/%s/%s/%s/
    %s'%(run_result_handle, packetlist_id, report_id,
    transaction_id, engine_id))
```

```
return
resp['successfullTransactionAnalysis'][0]['result']
```

AnalysisArtifact

A file found within a transaction, such as a picture, movie, doc, text, etc.

GET

`[base address]/shunra/api/analysis/artifact/{filehandle}`

Where the artifact handle is taken from the analysis report (See **Structure of an Analysis Report**).

http://localhost:8182/shunra/api/analysis/artifact/620984c9a31b4ef694a1ac47d61b6a7e%2F-561678026%2Fccb8713e522241c9a691c4ed1ce72d27%2F94660f9c01724f63bedfefb370dc4575%2Fabf8bde63762421dbe29cab1cecae661

Returns

- 200 "OK"
- 404 "Not Found"
- 500 "Internal Server Error"

Structure of an Analysis Report

As a result of executing step (4) "Get Analysis Result" for a specific transaction, the API returns a JSON document of the following format:

```
{
    "name": "Packet List 1",
    "successfullTransactionAnalysis": [{
        "status": "Finished",
        "result": {
            "type": "Best Practices Report",
            "subtype": "Web Applications Best Practices Report",
            "version": "0.5",
            -- Other, analysis engine dependent fields --
        },
        "name": "Undefined",
        "id": "fe15bdf3eafe4ec8bb1b055c49ca622b"
    }],
    "reportType": "best practices",
    "reportId": "129778102",
```

```
    "id": "1ae2d2ee02144e69801e0f3d1cb39d89",
    "failedTransactionAnalysis": []
}
```

Note: The text in blue will list the actual report. Notice that since the request is for a specific engine for a specific transaction, the results for `"successfullTransactionAnalysis"` contains only one entry, which lists the report for that transaction, wrapped with the transaction's name and ID. The reports all share a common structure, with `"type"`, `"subtype"` and `"version"` fields.

Structure of the HTTP Waterfall Analysis Report

A typical HTTP Waterfall report contains many entries in the "subTransactions" list, each is one of two possible types:

- **HTTP request/response**: contains a singles HTTP request coming either from a HTTP session or a decrypted HTTPS session
- **HTTPS session**: contains details about an non-decrypted HTTPS session (highlighted in red)

Not all fields are mandatory. Below, only the fields marked in red are guaranteed to be available in each entry. For example, if the response to a given request was not captured in the packet list, all the fields associated with a response do not appear in the entry. Thus, only the component that details the request's timestamps is guaranteed to be available. (Currently, the report contains only HTTP request/response pairs where the request was captured).

Timestamps are marked in blue and represent the number of seconds since January 1st, 1970. The handle to the response data is marked in orange. The response data itself can be retrieved by using the "Get Analysis Artifact" (5) API call.

```
{
  "type": "Waterfall report",
  "subtype": "Http Waterfall report",
  "version": "0.80",
  "subTransactions": [
    {
      "type": "HTTP request/response",
      "start": 1333054863953,
      "end": 1333054864640,
      "recomendations": "",
```

```
      "attributes": {
        "RequestContentSize": 0,
        "ResponseContentType": "application/json; char-
set\u003dUTF-8",
        "StatusCode": 401,
        "TcpReset": false,
        "Method": "POST",
        "Scheme": "https",
        "ResponseContentSize": 104,
        "RequestHeaders": "POST /setup/ws/1/validate HTTP/
1.1\r\nHost: setup.example.com\r\ nAccept-Encoding: gzip,
deflate\r\nConnection: keep-alive\r\nProxy-Connection: keep-
alive\r\n",
        "TcpSession": 4,
        "RequestData": "",
        "RequestContentType": "text/plain",
        "URI": "/setup/ws/1/validate",
        "ResponseData":
"\\74b85bbff75340a9b744bf8b4d1f5f6b\\-
1019702096\\5d35c39d1db84f3fa16786dc78eff622\\0703708ccbda491ba6
d59944c1ef1114/78820e83d8634265900999172f134389",
        "ResponseHeaders": "HTTP/1.1 401 Unauthorized\r\nDate:
Thu, 29 Mar 2012 21:01:04 GMT \r\nConnection: Keep-alive\r\n",
        "host": "setup.example.com",
        "Referer": "https://www.example.com/"
      },
      "components": [
        {
          "type": "DNSResolution",
          "start": 1333054863853,
          "end": 1333054863900
        },
        {
          "type": "TCPSetup",
          "start": 1333054863900,
          "end": 1333054863953
        },
        {
          "type": "ClientWaitAfterTCPSetup",
          "start": 1333054863953,
          "end": 1333054864418
        },
        {
          "type": "TLSHandshake",
```

```
        "start": 1333054864418,
        "end": 1333054864632
      },
      {
        "type": "request",
        "start": 1333054864632,
        "end": 1333054864632
      },
      {
        "type": "wait",
        "start": 1333054864632,
        "end": 1333054864640
      },
      {
        "type": "response",
        "start": 1333054864640,
        "end": 1333054864640
      }
    ]
  },

  {
    "type": "HTTPS session",
    "start": 1333054861902,
    "end": 1333054863281,
    "recomendations": "",
    "attributes": {
      "SentBytes": 384,
      "ReceivedBytes": 5792,
      "host": "www.example.com",
      "TcpReset": false,
      "TcpSession": 0
    },
    "components": [
      {
        "type": "TCPSetup",
        "start": 1333054861902,
        "end": 1333054861902
      },
      {
        "type": "ClientWaitAfterTCPSetup",
        "start": 1333054861902,
        "end": 1333054862731
      },
```

```
        {
          "type": "TLSHandshake",
          "start": 1333054862731,
          "end": 1333054863230
        },
        {
          "type": "EncryptedDataTransmission",
          "start": 1333054863230,
          "end": 1333054863281
        }
      ]
    }
       --- OTHER HTTP and HTTPS ENTRIES --

  ]
}
```

Structure of the Best Practices Analysis Report

The best practices report is quite simple. `report` is a list of entries, each representing a best practice; In the example below two best practices are highlighted in blue. Each best practice contains the following fields:

- **Name**
- **Description**
- **List of applicable scenarios** (currently DesktopWeb or MobileSafari)
- **Score**: which measures the how much the transaction follows the given best practice (a number in [0,1]),
- **Weight**: measures the impact of the best practice on the transaction (a number in [0,1]).
- **A dictionary of violations**: each entry in this dictionary is a specific type of violation on the best practice and a list of resources (or TCP sessions) that are committing that violation. Notice that a transaction may not have any violation for a given best practice, as is the case with `"Compress Components"` below.

```
{
  "type": "Best Practices Report",
  "subtype": "Web Applications Best Practices Report",
  "version": "0.5",
  "report": [
    {
      "violations": {},
```

```
      "name": "Compress Components",
      "scenarios": [
        "DesktopWeb",
        "MobileSafari"
      ],
      "description": "Checks that textual elements are trans-
ferred in a compressed format. Compression usually reduces the
response size by about 70%. Approximately 90% of current Internet
traffic travels through browsers that claim to support gzip. ",
      "score": 100.0,
      "weight": 1.0
    },

    {
      "violations": {
        "An expiration header was not found": [
          "http://platform.example.com/widgets.js"
        ],
        "Expiration date is within the next two days": [
          "HTTP://media.example.com/media-proxy/picture1.jpg",
      "http://media.example.com/media-proxy/picture2.jpg",

                "http://media.example.com/media-proxy/
picture3.jpg"
        ]
      },
      "name": "Add long term headers expiration dates",
      "scenarios": [
        "DesktopWeb",
        "MobileSafari"
      ],
      "description": "Near future headers expiration dates pre-
vent effective caching. This results in a repeat visit to your
site to be slower than necessary.",
      "score": 65.0,
      "weight": 0.8
    }
  ]
}
```

4

Shunra Analytics Protocols

This section provides details regarding how Shunra Analytics identifies and works with various protocols in the legacy reports, including:

Supported Protocols

The following protocols are supported and analyzed by Shunra Analytics.

Layer 2 - 3	Web
IP	HTTP
TCP	HTTPS
UDP	

Understanding Protocol Association

Conversations are associated based on the relevant application protocol. This means that if we're looking at HTTP then the underlying TCP / IP communication and communication metrics will be associated with

the HTTP conversation they are part of. If we found TCP Retransmissions during a HTTP Get Request-Response conversation, these TCP Retransmissions will be associated with the HTTP conversation.

When a sequence number is received that is lower than expected (i.e. either a retransmission, a fast retransmission, or an out of order segment), Shunra Analytics assumes that it is a fast retransmission if:

- It has seen >= 2 duplicate ACKs for this segment (i.e. >=3 ACKs).
- If this segment is the next un-ACKed segment.
- If this segment came within 20ms of the last duplicate ACK (20ms is arbitrary; it should be small enough to not be confused with a retransmission timeout).

Sub-Transaction Grouping

The Shunra Analytics groups sub-conversations into a single flow so that you can get data of the whole conversation or on each of the sub-conversations that it contains. How this grouping takes places is determined by how you configure Shunra Analytics for grouping.

Classification of TCP, UDP, IP

A conversation is classified as TCP only if no higher level protocol is present. In this case it is identified as TCP Other.

A conversation is classified as UDP only if no higher level protocol is present. In this case it is identified as UDP Other.

A conversation is classified as IP only if there is no higher level protocol e.g. not TCP or UDP). In this case it is identified as IP Other.

Collecting Conversation Statistics

Shunra Analytics collects statistics per conversation instance (e.g. a single Get of www.google.com URL).

Metrics shown will be for the following groupings:

- All applications
- Per application
- Per application conversation (including sub-conversation)

Conversation Definition

The definition and identification of a conversation depends on the type of analysis being performed.

The definitions (identifications) are based on:

- IP - IP address pair (e.g. 10.0.0.1 - 10.0.0.2)
- UDP - IP address & port number pair (e.g. 10.0.0.1:6789 - 10.0.0.2:3456)
- TCP - IP address & port number pair (e.g. 10.0.0.1:6789 - 10.0.0.2:3456)
- HTTP - URL (e.g. www.google.com/images)