EventTracker

Actionable Security Intelligence

Integrate Mimecast Secure Email Gateway

EventTracker v8.x and above

Publication Date: January 5, 2018

Abstract

This guide provides instructions to configure Mimecast Secure Email Gateway to send crucial events to EventTracker Enterprise by means of syslog.

Scope

The configurations detailed in this guide are consistent with **EventTracker Enterprise version 8.x and later**, and **Mimecast Secure Email Gateway latest version.**

Audience

Mimecast Secure Email Gateway users, who wish to forward its events to EventTracker Manager and monitor them using EventTracker Enterprise.

Table of Contents

Overview

Mimecast Secure Email Gateway combines strong defenses to keep sophisticated attackers out, and innovative applications and policies to keep sensitive information secure. Multi-layered detection engines and intelligence protect the company and employees from spear-phishing, malware, spam and zero-day attacks.

EventTracker collects and analyses mail gateway events and enlightens an administrator about security violations, spam emails, and email traffic anomalies.

Prerequisites

- **EventTracker** should be installed.
- **Mimecast Secure Email Gateway** latest version should be installed.
- **Python 3.0 and above** should be installed.
- **Windows PowerShell** version 4 and above.
- If port **514** is disabled by firewall it needs to be enabled for syslog forwarding.

Enable Syslog Forwarding in Mimecast Secure Email Gateway

Preparation Steps

This sample script requires the Access Key and Secret Key from a Mimecast Authentication token for a Mimecast administrator. By default, an Authentication Token expires after 3 days, this means that your script would stop being able to collect data and events after 3 days without manual intervention.

Consequently, for the best experience you must create a new user and Authentication Profile defining a longer-lived Authentication Token. The steps below describe this process:

Step 1: Create a new user

- Login to the Administration Console.
- Navigate to the **Administration | Directories | Internal Directories** menu item to display a list of internal domains.
- Select the internal domain where you would like to create your new user.
- Select the **New Address** button from the menu bar.
- Complete the new address form and select **Save and Exit** to create the new user.
- Keep a note of the password set as you will use this to get your Authentication Token in Step 6.

Step 2: Add the user to an Administrative Role

- While logged into the Administration Console, navigate to the **Administration | Account | Roles** menu item to display the Roles page.
- Right click the **Basic Administrator** role and select **Add users to role**.
- Browse or search to find the new user created in the Step 1.
- Select the tick box to the left of the user.
- Select the **Add selected users** button to add the user to the role.

Step 3: Create a new group and add your new user

- While logged into the Administration Console, navigate to **the Administration | Directories | Profile Groups** menu item to display the Profile groups page.
- Create a new group by selecting the plus icon on the parent folder where you would like to create the group. This creates a new group with the Name "New Folder"
- To rename the group, select the newly created "New Folder" group. Then from the **Edit group** text box type the name you want to give the folder, for example SIEM Admin and press the Enter key to apply the change.
- With the group selected select the Build drop down button and select **Add Email Addresses**.
- Type the name of the new user created in Step 1.
- Select **Save and Exit** to add the new user to the group.

Step 4: Create a new Authentication Profile

- While logged into the Administration Console, navigate to the **Administration | Services | Applications menu** item to display the Application Settings page.
- Select the **Authentication Profiles** button.
- Select the **New Authentication Profile** button.
- Type a **Description** for the new profile.

- Set the **Authentication TTL** setting to **Never Expires**. This will make sure that when you create your Authentication Token it will not expire and impact the data collection of the app.
- Leave all other settings as their default.
- Select **Save and Exit** to create the profile.

Step 5: Create a new Application Setting

- While logged into the Administration Console, navigate to **the Administration | Services | Applications** menu item to display the Application Settings page.
- Select the **New Application Settings** button.
- Type a **Description**.
- Use the Group **Lookup** button to select the **Group** that you created in Step 3.
- Use the Authentication Profile **Lookup** button to select the **Authentication Profile** created in Step 4.
- Leave all other settings as their default.
- Select **Save and Exit** to create and apply the Application Settings to your new group and user.

Step 6: (a).Get your authentication token

Now that you have a dedicated administrator to create an Authentication Token that will never expire, the final preparation task is to get the Authentication Token for the user.

Getting an Authentication token using Mac OSX or *nix systems.

1. Open a terminal application and type the following command to generate a base64 encoded string of your administrators email address and password:

   ```
   echo -n 'email_address:password' | openssl base64
   ```

   Where email_address is the email address of the user created in **Step 1** and password is the password created for the user in **Step 1**. Be sure to include the ":" between the email_address and password as authentication will fail without it.
2. Type the following command to use cURL to login to the Mimecast API and get your Authentication Token.

   ```
   curl -i -H 'Authorization: Basic-Cloud base64_encoded_username_password' -H 'x-mc-app-id: YOUR APPLICATION ID' -H 'Content-Type:application/json' https://xx-api.mimecast.com/api/login/login --data-binary '{"data":[{"username": "email_address"}]}'
   ```

   ```
   curl -i -H 'Authorization: Basic-Cloud base64_encoded_username_password' -H 'x-mc-app-id: 1f3287ec-4e7c-11e6beb8-9e71128cae77' -H 'Content-Type:application/json' https://xx-api.mimecast.com/api/login/login --data-binary '{"data":[{"username": "email_address"}]}'
   ```

Where:

base64_encoded_username_password is the value generated in step 1.
x-mc-app-id is the value of your application id.
xx-api is the base url for the region where your Mimecast account is hosted as documented in the System Requirements section.
email_address is the email address of the user created in **Step 1: Create a new user**.

3. An example response to this command is:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-control: no-store
Pragma: no-cache
Content-Length: 375
Content-MD5: 124911b164dbd3b9e823610a2eb4996a
Date: Mon, 25 Jul 2016 16:19:37 +0100
Connection: Keep-Alive
{"meta":{"status":200},"data":
[{"accessKey":"LOW__TRUNCATED__2nN","secretKey":"jD9__TRUNCATED__gIEJdC4e/Q\u003d
\u003d","duration":3153600000000,"bindingType":"one_step","extendOnValidate":false}],"fail": []}
```

Copy and paste the accessKey and secretKey values from the response to your script in the ACCESS_KEY and SECRET_KEY variables respectively.

4. **IMPORTANT**: make sure to replace the \u003d\u003d at the end of the secret key with == (\u003d is the uri encoding for the = symbol and is printed to the terminal, however the actual string should contain the = symbol when used)

(b).Getting an Authentication token using Windows PowerShell

1. Copy paste the following script into a PowerShell window:

```
$creds = Get-Credential

$discoverPostBody = @{"data" = ,@{"emailAddress" = $creds.UserName}}

$discoverPostBodyJson = ConvertTo-Json $discoverPostBody

$discoverRequestId = [guid]::NewGuid().guid

$discoverRequestHeaders = @{"x-mc-app-id" = "1f3287ec-4e7c-11e6-beb8-9e71128cae77"; "x-mc-req-id" = $discoverRequestId; "Content-Type" = "application/json"}

$discoveryData = Invoke-RestMethod -Method Post -Headers $discoverRequestHeaders -Body $discoverPostBodyJson -Uri "https://api.mimecast.com/api/login/discover-authentication"
```

```
$baseUrl = $discoveryData.data.region.api

$keys = @{}

$uri = $baseUrl + "/api/login/login"

$requestId = [guid]::NewGuid().guid

$netCred = $creds.GetNetworkCredential()

$PlainPassword = $netCred.Password

$credsBytes = [System.Text.Encoding]::ASCII.GetBytes($creds.UserName + ":" + $PlainPassword)

$creds64 = [System.Convert]::ToBase64String($credsBytes)

$headers = @{"Authorization" = "Basic-Cloud " + $creds64; "x-mc-app-id" = "1f3287ec-4e7c-11e6beb8-9e71128cae77"; "x-mc-req-id" = $requestId; "Content-Type" = "application/json"}

$postBody = @{"data" = ,@{"username" = $creds.UserName}}

$postBodyJson = ConvertTo-Json $postBody

$data = Invoke-RestMethod -Method Post -Headers $headers -Body $postBodyJson -Uri $uri

"Access key: " + $data.data.accessKey

"Secret key: " + $data.data.secretKey
```

2. Enter the email address and password of the user created in **Step 1: Create a new user** into the Windows credentials box that will launch after you have pasted the script into the PowerShell window.
3. Copy and paste the accessKey and secretKey values printed at the bottom of the PowerShell window to your script in the ACCESS_KEY and SECRET_KEY variables respectively.

Step 7: Enable logging for your account

- While logged into the Administration Console, navigate to **the Administration | Account | Account Settings** menu item to display the Account Settings page.
- Select the **Enhanced Logging** section.
- Select the types of logs you want to enable. The choices are:
  - **Inbound** - logs for messages from external senders to internal recipients
  - **Outbound** - logs for messages from internal senders to external recipients
  - **Internal** - logs for messages between internal domains
- Select **Save** to apply the changes.

Once these settings have been saved the Mimecast MTA will start logging data for your account and logs should start to become available for download up to 30 minutes after that.

Using the script

- Copy the below script to a text editor and save the file with a .py file extension.
- Edit the #**Set up variables** section.
  - Add your Application ID and Application Key. Please see the **Getting Started with the Mimecast API guide** for details on getting these values if you do not have them already.
  - Add your Mimecast administrator's email address.
  - Add a fully qualified path to the folders to be used to write the log files and page tokens. Make sure that the user that will be running this script has permission to write to the folder.
- Save the file.
- The script should be ready to be executed.

Python Script

```
#!/usr/bin/env python
# -*- coding: utf-8 -*
import logging
import logging.handlers
import json
import os
import requests
import base64
import uuid
import datetime
import hashlib
import hmac
import time
# Set up variables
APP_ID = "YOUR DEVELOPER APPLICATION ID"
APP_KEY = "YOUR DEVELOPER APPLICATION KEY"
URI = "/api/audit/get-siem-logs"
EMAIL_ADDRESS = 'EMAIL ADDRESS OF YOUR ADMINISTRATOR'
```

```
ACCESS_KEY = 'ACCESS KEY FOR YOUR ADMINISTRATOR'
SECRET_KEY = 'SECRET KEY FOR YOUR ADMINISTRATOR'
LOG_FILE_PATH = "FULLY QUALIFIED PATH TO FOLDER TO WRITE LOGS"
CHK_POINT_DIR = 'FULLY QUALIFIED PATH TO FOLDER TO WRITE PAGE TOKEN'
# Set True to output to syslog, false to only save to file
syslog_output = True
# Enter the IP address or hostname of your syslog server
syslog_server = '127.0.0.1'
# Change this to override default port
syslog_port = 514
# Set up logging (in this case to terminal)
log = logging.getLogger(__name__)
log.root.setLevel(logging.DEBUG)
log_formatter = logging.Formatter('%(levelname)s %(message)s')
log_handler = logging.StreamHandler()
log_handler.setFormatter(log_formatter)
log.addHandler(log_handler)
# Set up syslog output
syslog_handler = logging.handlers.SysLogHandler(address=(syslog_server, syslog_port))
syslog_formatter = logging.Formatter('%(message)s')
syslog_handler.setFormatter(syslog_formatter)
syslogger = logging.getLogger(__name__)
syslogger = logging.getLogger('SysLogger')
syslogger.addHandler(syslog_handler)
# Supporting methods
def get_hdr_date():
    date = datetime.datetime.utcnow()
```

```
    dt = date.strftime("%a, %d %b %Y %H:%M:%S")
    return dt + " UTC"
def read_file(file_name):
    try:
        with open(file_name, 'r') as f:
            data = f.read()
        return data
    except Exception, e:
        log.error('Error reading file ' + file_name + '. Cannot continue. Exception: ' + str(e))
        quit()
def write_file(file_name, data_to_write):
    try:
        with open(file_name, 'w') as f:
            f.write(data_to_write.encode('utf-8'))
    except Exception, e:
        log.error('Error writing file ' + file_name + '. Cannot continue. Exception: ' + str(e))
        quit()
def delete_file(file_name):
    try:
        os.remove(file_name)
    except Exception, e:
        log.error('Error deleting file ' + file_name + '. Cannot continue. Exception: ' + str(e))
        quit()
def create_signature(data_to_sign, secret_key):
    digest = hmac.new(secret_key.decode("base64"), data_to_sign, digestmod=hashlib.sha1).digest()
    return base64.encodestring(digest).rstrip()
def get_base_url(email_address):
```

```
  # Create post body for request
  post_body = dict()
post_body['data'] = [{}]
  post_body['data'][0]['emailAddress'] = email_address
  # Create variables required for request headers
  request_id = str(uuid.uuid4())
  request_date = get_hdr_date()
  headers = {'x-mc-app-id': APP_ID, 'x-mc-req-id': request_id, 'x-mc-date': request_date}
  # Send request to API
  log.debug('Sending request to https://api.mimecast.com/api/discover-authentication with request Id: ' +
            request_id)
  try:
    r = requests.post(url='https://api.mimecast.com/api/login/discover-authentication',
                      data=json.dumps(post_body), headers=headers)
    # Handle Rate Limiting
    if r.status_code == 429:
      log.warn('Rate limit hit. sleeping for ' + str(r.headers['X-RateLimit-Reset'] * 1000))
      time.sleep(r.headers['X-RateLimit-Reset'] * 1000)
  except Exception, e:
    log.error('Unexpected error getting base url. Cannot continue.' + str(e))
    quit()
  # Handle error from API
  if r.status_code != 200:
    log.error('Request returned with status code: ' + str(r.status_code) + ', response body: ' +
              r.text + '. Cannot continue.')
    quit()
  # Load response body as JSON
```

```
    resp_data = json.loads(r.text)
    # Look for api key in region region object to get base url
    if 'region' in resp_data["data"][0]:
        base_url = resp_data["data"][0]["region"]["api"].split('//')
        base_url = base_url[1]
    else:
        # Handle no region found, likely the email address was entered incorrectly
        log.error(
            'No region information returned from API, please check the email address.'
 'Cannot continue')
        quit()
    return base_url
def post_request(base_url, uri, post_body, access_key, secret_key):
    # Create variables required for request headers
    request_id = str(uuid.uuid4())
    request_date = get_hdr_date()
    signature = 'MC ' + access_key + ':' + create_signature(':'.join(
        [request_date, request_id, uri, APP_KEY]), secret_key)
    headers = {'Authorization': signature, 'x-mc-app-id': APP_ID, 'x-mc-req-id': request_id, 'x-mc-date':
request_date}
    try:
        # Send request to API
        log.debug('Sending request to https://' + base_url + uri + ' with request Id: ' + request_id)
        r = requests.post(url='https://' + base_url + uri, data=json.dumps(post_body), headers=headers)
        # Handle Rate Limiting
        if r.status_code == 429:
            log.warn('Rate limit hit. sleeping for ' + str(r.headers['X-RateLimit-Reset'] * 1000))
```

```
            time.sleep(r.headers['X-RateLimit-Reset'] * 1000)
            r = requests.post(url='https://' + base_url + uri, data=json.dumps(post_body), headers=headers)
    # Hamdle errors
    except Exception, e:
        log.error('Unexpected error connecting to API. Exception: ' + str(e))
        return 'error'
    # Handle errors from API
    if r.status_code != 200:
        log.error('Request to ' + uri + ' with , request id: ' + request_id + ' returned with status code: ' +
                  str(r.status_code) + ', response body: ' + r.text)
        return 'error'
    # Return response body and response headers
    return r.text, r.headers
def get_mta_siem_logs(checkpoint_dir, base_url, access_key, secret_key):
    uri = "/api/audit/get-siem-logs"
    # Set checkpoint file name to store page token
    checkpoint_filename = os.path.join(checkpoint_dir, 'get_mta_siem_logs_checkpoint')
    # Build post body for request
    post_body = dict()
    post_body['data'] = [{}]
    post_body['data'][0]['type'] = 'MTA'
    if os.path.exists(checkpoint_filename):
        post_body['data'][0]['token'] = read_file(checkpoint_filename)
    # Send request to API
    resp = post_request(base_url, uri, post_body, access_key, secret_key)
    # Process response
    if resp != 'error':
```

```
resp_body = resp[0]
resp_headers = resp[1]
content_type = resp_headers['Content-Type']
# End if response is JSON as there is no log file to download
if content_type == 'application/json':
    log.info('No more logs available')
    return False
# Process log file
elif content_type == 'application/octet-stream':
    file_name = resp_headers['Content-Disposition'].split('=\"')
    file_name = file_name[1][:-1]
    # Save file to LOG_FILE_PATH
    write_file(os.path.join(LOG_FILE_PATH, file_name), resp_body)
    # Save mc-siem-token page token to check point directory
    write_file(checkpoint_filename, resp_headers['mc-siem-token'])
    try:
        if syslog_output is True:
            log.info('Loading file: ' + os.path.join(LOG_FILE_PATH, file_name) + ' to output to ' +
                                        syslog_server + ':' + str(syslog_port))
            with open(os.path.join(LOG_FILE_PATH, file_name), 'r') as log_file:
                lines = log_file.read().splitlines()
                for line in lines:
                    syslogger.info(line)
            log.info('Syslog output completed for file ' + file_name)
    except Exception, e:
        log.error('Unexpected error writing to syslog. Exception: ' + str(e))
```

```
            # return true to continue loop
            return True
        else:
            # Handle errors
            log.error('Unexpected response')
            for header in resp_headers:
                log.error(header)
            return False
def run_script():
    # discover base URL
    try:
        base_url = get_base_url(email_address=EMAIL_ADDRESS)
    except Exception, e:
        log.error('Error discovering base url for ' + EMAIL_ADDRESS + ' . Exception: ' + str(e))
        quit()
    # Request log data in a loop until there are no more logs to collect
    try:
        log.info('Getting MTA log data')
        while get_mta_siem_logs(checkpoint_dir=CHK_POINT_DIR, base_url=base_url, access_key=ACCESS_KEY,
                                secret_key=SECRET_KEY) is True:
            log.info('Getting more MTA log files')
    except Exception, e:
        log.error('Unexpected error getting MTA logs ' + (str(e)))
    quit()
# Run script
run_script()
```

In the above script attached, the highlighted fields need to be provided with adequate credentials as shown below:

- **APP_ID =** "YOUR DEVELOPER APPLICATION ID"
- **APP_KEY =** "YOUR DEVELOPER APPLICATION KEY"
- **EMAIL_ADDRESS** = 'EMAIL ADDRESS OF YOUR ADMINISTRATOR'
- **ACCESS_KEY** = 'ACCESS KEY FOR YOUR ADMINISTRATOR'
- **SECRET_KEY** = 'SECRET KEY FOR YOUR ADMINISTRATOR'
- **LOG_FILE_PATH** = "FULLY QUALIFIED PATH TO FOLDER TO WRITE LOGS"
- **CHK_POINT_DIR** = 'FULLY QUALIFIED PATH TO FOLDER TO WRITE PAGE TOKEN'
- **Syslog_Server** = 'EventTracker Manager IP Address'
- **Syslog_port** = 514

EventTracker Knowledge Pack

Once logs are received into EventTracker, Categories and reports can be configured into EventTracker.

The following Knowledge Packs are available in EventTracker Enterprise to support Windows.

Alerts

- **Mimecast: Virus signature detection:** This alert is generated when a virus signature email is detected by the Mimecast email gateway.

Flex Reports

- **Mimecast-Inbound and outbound accepted emails -** This report provides details about all the inbound and outbound emails that are monitored by the Mimecast secure email gateway.

LogTime	Computer	Sender IP Address	Sender Address	Recipient Address	Subject	Direction	Message ID	Deliver Status	Recipient Acknowledgement	Attachment Size	Sent Bytes	Attempts	Latency	Mail Route
12/01/2017 01:55:01 PM	MIMECAST	220.17.30.11	MELGIBS@acme.org.org	scarlett@contoso.com	Network capable terminal	Inbound	5BB94EDF73A11246BAB5EECF3326B23B019479B79B@KJCUIFJ659.SUPPLY1.apps	true	\250 Ok: queued as 181A073801\	0	10164	1	4834	Inbound IP Route

Logs Considered:

LOG TIME	EVENT ID	SITE / COMPUTER	USER	DOMAIN	SOURCE
12/1/2017 1:55:01 PM	3333	NTPLDTBLR38 / Mimeca...	N/A	N/A	Syslog

Event Type: Information
Log Type: Application
Category Id: 0

Description:
Nov 27 09:11:34 w7-mime datetime=2017-11-20T09:40:56-0500|aCode=O2nICgFcN9SHwds0EmfMvg|acc=DBZ1425987|Delivered=true|IP=220.17.30.11|AttCnt=0|Dir=Inbound|ReceiptAck=\250 Ok: queued as 181A073801\|MsgId=<5BB94EDF73A11246BAB5EECF3326B23B019479B79B@KJCUIFJ659.SUPPLY1.apps>|Subject=\Network capable terminal\|Latency=4834|Sender=MELGIBS@acme.org.org|Rcpt=scarlett@contoso.com|AttSize=0|Attempt=1|TlsVer=TLSv1.2|Cphr=TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384|Snt=10164|UseTls=Yes|Route=\Inbound IP Route\

- **Mimecast-Rejected emails -** This report provides details about all the emails that are rejected by the Mimecast secure email gateway.

LogTime	Computer	Sender IP Address	Sender Address	Recipient Address	Subject	Message ID	Direction	Reject Type	Reject Information	Error Description
12/01/2017 01:55:01 PM	MIMECAST	172.168.10.4	zinaidadsfqz@spazioalpha.com.br	jsonmick@contoso.com	hello all	4136653vdgg66ag8g33sdbg9a	Inbound	Envelope Rejected	Administrative prohibition - envelope blocked	Policy level block list in force

Logs Considered:

LOG TIME	EVENT ID	SITE / COMPUTER	USER	DOMAIN	SOURCE
12/1/2017 1:55:01 PM	3333	NTPLDTBLR38 / Mimeca...	N/A	N/A	Syslog

Event Type: Information
Log Type: Application
Category Id: 0

Description:
Nov 27 09:11:04 w7-mime datetime=2017-11-20T09:38:19-0500|aCode=oQVDekF3NFC_39CSR4Qbrw|acc=DBZ1425987|Sender=zinaidadsfqz@spazioalpha.com.br|Rcpt=jsonmick@contoso.com|Act=Rej|RejInfo=\Administrative prohibition - envelope blocked\|IP=172.168.10.4|RejType=\Envelope Rejected\|Error=\Policy level block list in force\|RejCode=550|Dir=Inbound|headerFrom=

- **Mimecast-Spam emails -**This report provides details about all the spam emails that are detected by the Mimecast secure email gateway.

LogTime	Computer	Sender IP Address	Sender Address	Recipient Address	Subject	Message ID	Direction	Reject Information	Spam Info	Error Description	Spam Limit	Spam Score
12/01/2017 01:55:01 PM	MIMECAST	230.14.23.131	bounce-1984760-1072064-1663052-20240@vvt.com	jreacher@contoso.com	Re: LIGHTING LED	d73867710c300568ff315eaf61e13df2@15.22.4.19	Inbound	[MCSpamSignature.r.s.30.458]|SpamScore=30	[MCSpamSignature.r.s.30.458]	[MCSpamSignature.r.s.30.458]	28	30

Logs Considered:

LOG TIME	EVENT ID	SITE / COMPUTER	USER	DOMAIN	SOURCE
12/1/2017 1:55:01 PM	3333	NTPLDTBLR38 / Mimeca...	N/A	N/A	Syslog

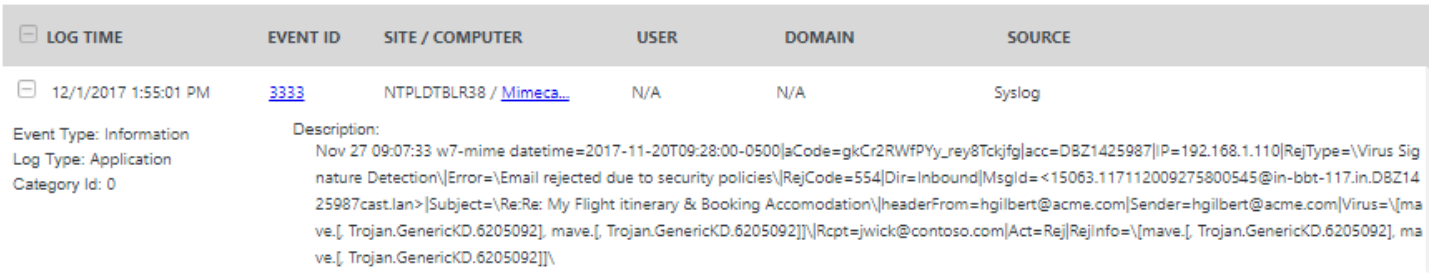
Event Type: Information
Log Type: Application
Category Id: 0

Description:
Nov 27 09:10:41 w7-mime datetime=2017-11-20T09:45:57-0500|aCode=Y8q0SxQ6OXieZ4TafANBUQ|acc=DBZ1425987|SpamLimit=28|IP=230.14.23.131|Error=[MCSpamSignature.r.s.30.458]|RejCode=554|Dir=Inbound|MsgId=<d73867710c300568ff315eaf61e13df2@15.22.4.19>|Subject=\Re: LIGHTING LED\|headerFrom=ravensp@acme.com|Sender=bounce-1984760-1072064-1663052-20240@vvt.com|Rcpt=jreacher@contoso.com|SpamInfo=[MCSpamSignature.r.s.30.458]|Act=Rej|RejInfo=[MCSpamSignature.r.s.30.458]|SpamScore=30

- **Mimecast-Virus signature detection -** This report provides details about all the emails that contain a virus signature or a suspicious phishing details.

LogTime	Computer	Sender IP Address	Sender Address	Recipient Address	Subject	Message ID	Direction	Reject Type	Reject Information	Virus Details	Error Description
12/01/2017 01:55:01 PM	MIMECAST	192.168.1.110	hgilbert@acme.com	jwick@contoso.com	Re:Re: My Flight itinerary & Booking Accomodation	15063.117112009275800545@in-bbt-117.in.DBZ1425987cast.lan	Inbound	Virus Signature Detection	[mave.[,Trojan.GenericKD.6205092], mave.[,Trojan.GenericKD.6205092]]	Email rejected due to security policies	Email rejected due to security policies

Logs Considered:



LOG TIME	EVENT ID	SITE / COMPUTER	USER	DOMAIN	SOURCE
12/1/2017 1:55:01 PM	3333	NTPLDTBLR38 / Mimeca...	N/A	N/A	Syslog

Event Type: Information
Log Type: Application
Category Id: 0

Description:
Nov 27 09:07:33 w7-mime datetime=2017-11-20T09:28:00-0500|aCode=gkCr2RWfPYy_rey8Tckjfg|acc=DBZ1425987|IP=192.168.1.110|RejType=\Virus Signature Detection\|Error=\Email rejected due to security policies\|RejCode=554|Dir=Inbound|MsgId=<15063.117112009275800545@in-bbt-117.in.DBZ1425987cast.lan>|Subject=\Re:Re: My Flight itinerary & Booking Accomodation\|headerFrom=hgilbert@acme.com|Sender=hgilbert@acme.com|Virus=\[mave.[, Trojan.GenericKD.6205092], mave.[, Trojan.GenericKD.6205092]]\|Rcpt=jwick@contoso.com|Act=Rej|RejInfo=\[mave.[, Trojan.GenericKD.6205092], mave.[, Trojan.GenericKD.6205092]]\

Import Mimecast Secure Email Gateway Knowledge Pack into EventTracker

NOTE: Import knowledge pack items in the following sequence:

- Knowledge Objects
- Categories
- Alerts
- Parsing Rules
- Flex Reports

NOTE: Export knowledge pack items in the following sequence:

- Knowledge Objects
- Categories
- Alerts
- Parsing Rules
- Flex Reports

1. Launch **EventTracker Control Panel**.

2. Double click **Export Import Utility**, and then click the **Import** tab.

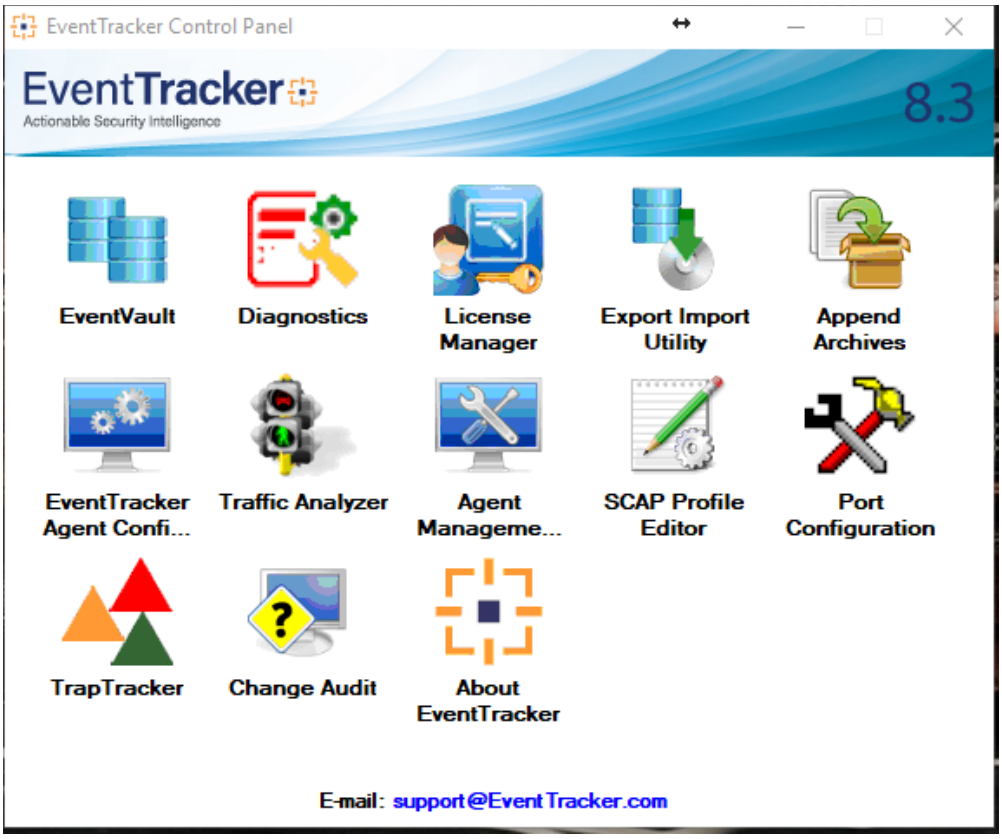


Figure 2

Import Alerts

1. Click **Alert** option, and then click the **browse**  button.

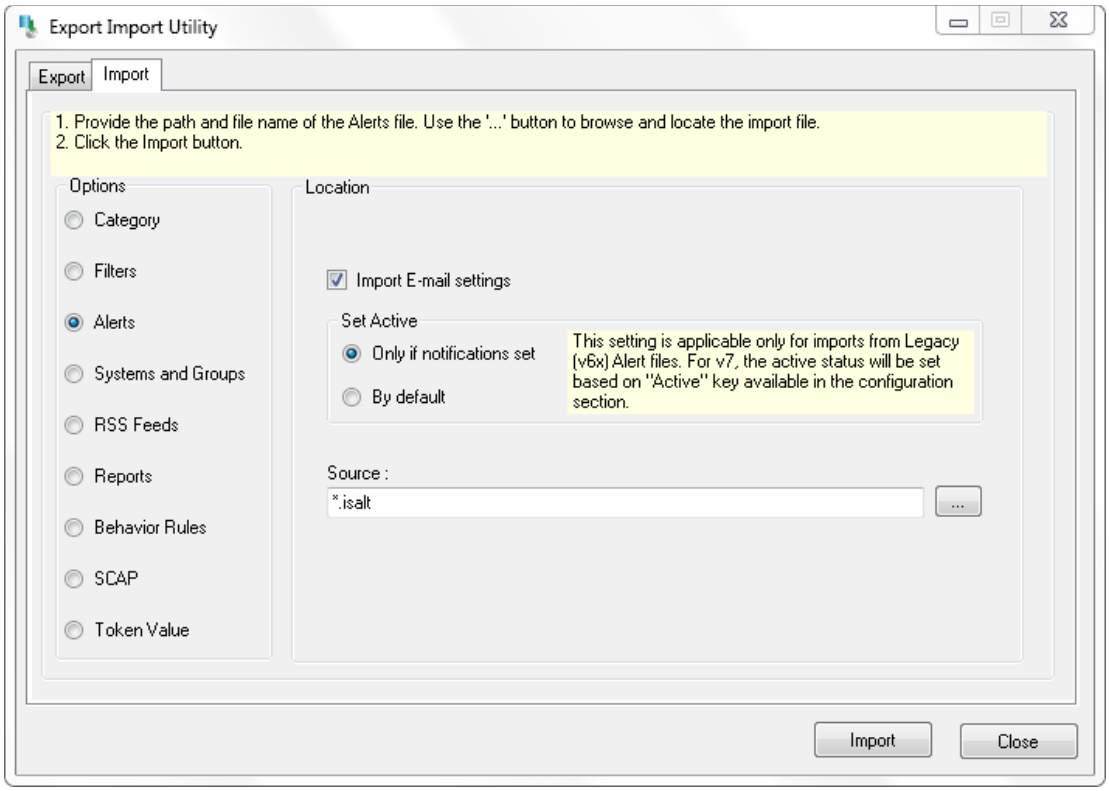


Figure 3

2. Locate **Mimecast alerts.isalt** file, and then click the **Open** button.
3. To import alerts, click the **Import** button.
4. EventTracker displays success message.

Figure 3

5. Click the **OK** button, and then click the **Close** button.

Import Knowledge Object

1. Click the **Admin** menu, and then click **Knowledge Objects**.
2. Click on ‘**Import**’ option.

Figure 4

3. In **IMPORT** pane click on **Browse** button.

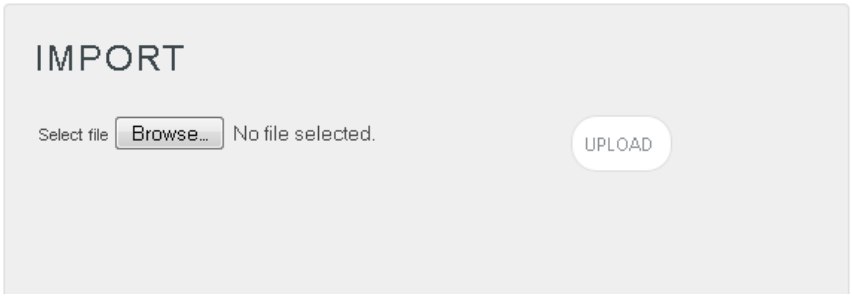


Figure 5

4. Locate **Mimecast knowledge objects.etko** file, and then click the **UPLOAD** button.

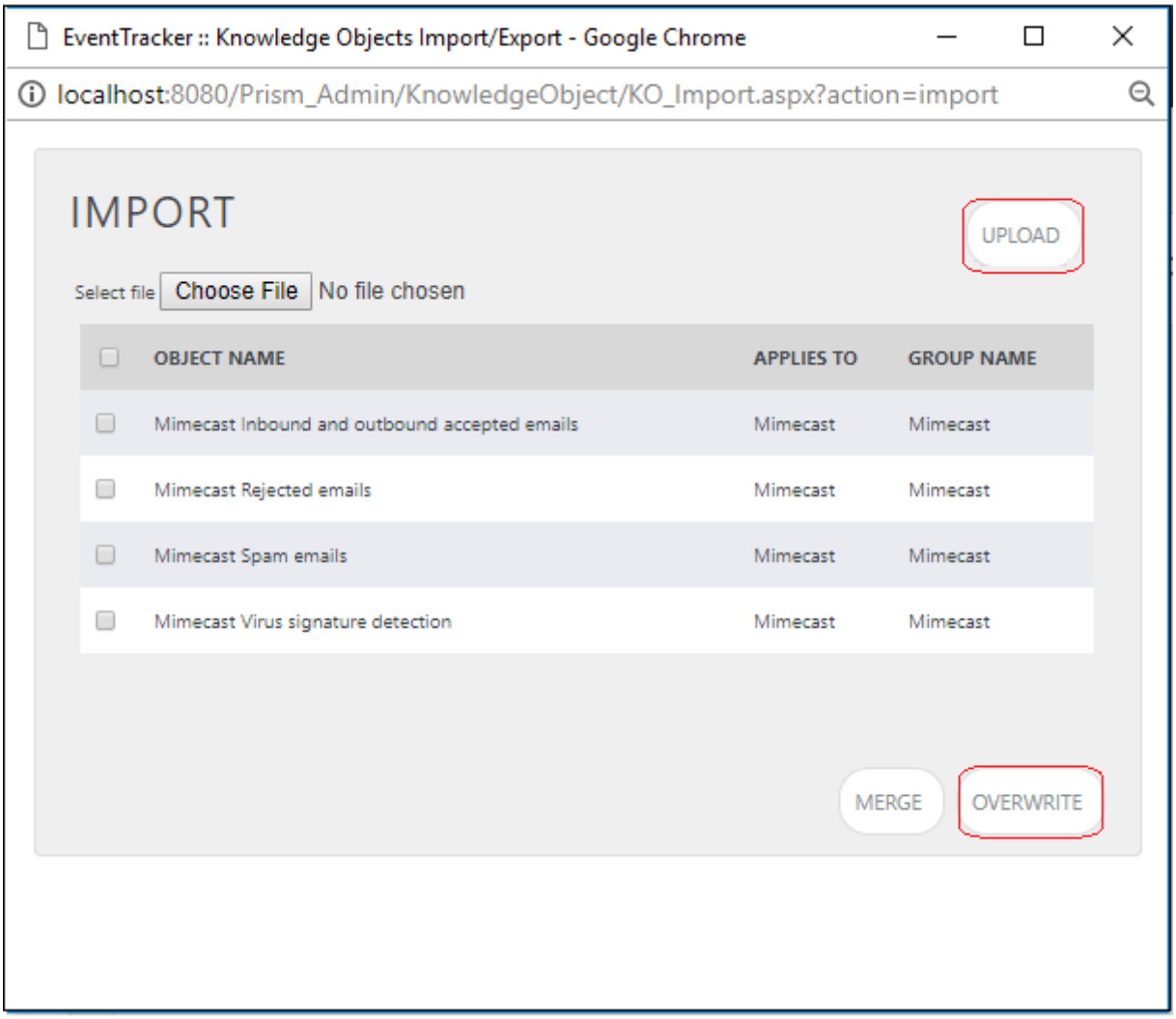


Figure 6

5. Now select the check box and then click on '**OVERWRITE**' option. EventTracker displays success message.

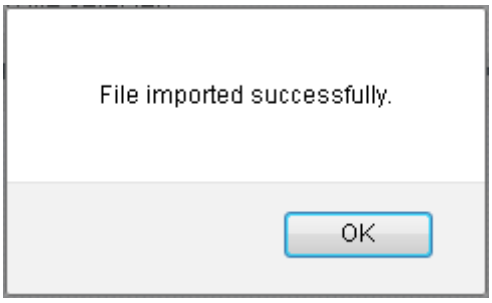


Figure 7

6. Click on **OK** button.

Parsing Rule

1. Click **Token Value** option, and then click the browse button.

2. Locate the **Mimecast Tokens.istoken** file, and then click the **Open** button.

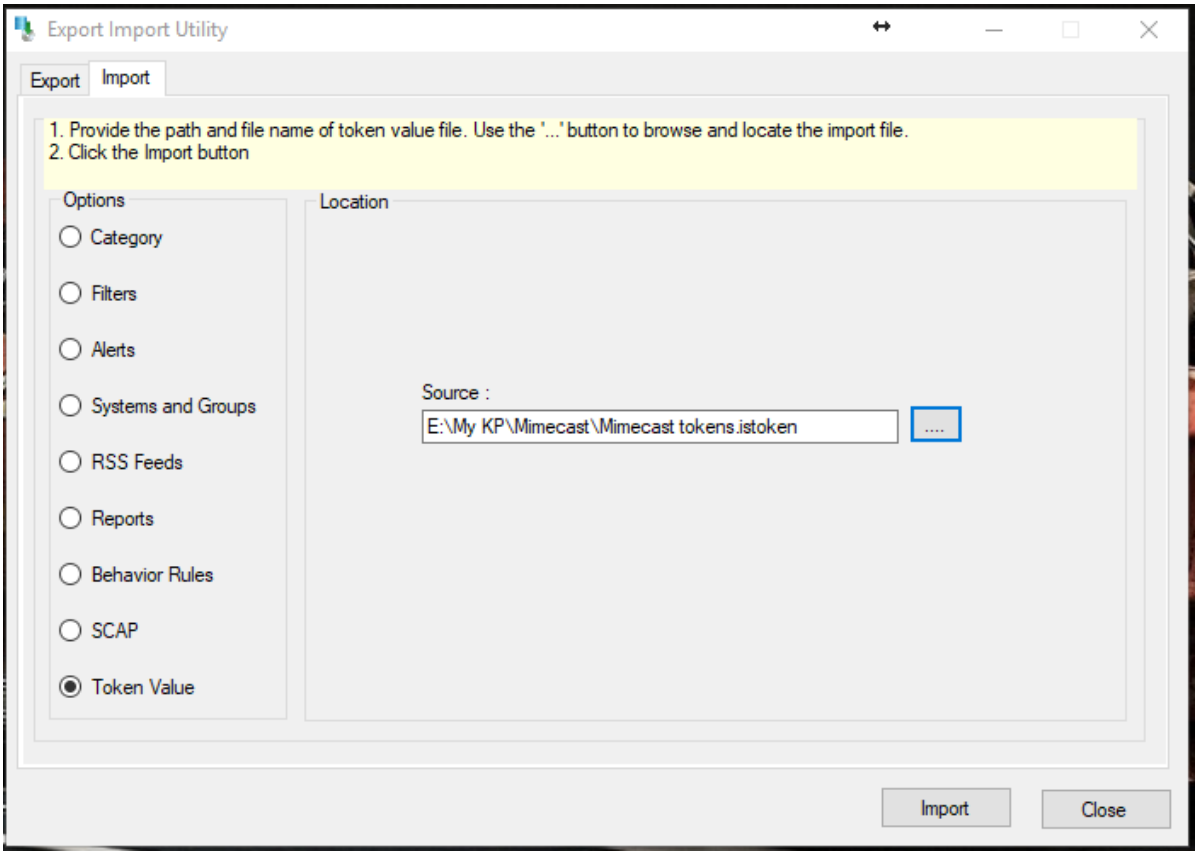


Figure 8

1. Click the **Import** button to import the tokens. EventTracker displays success message.

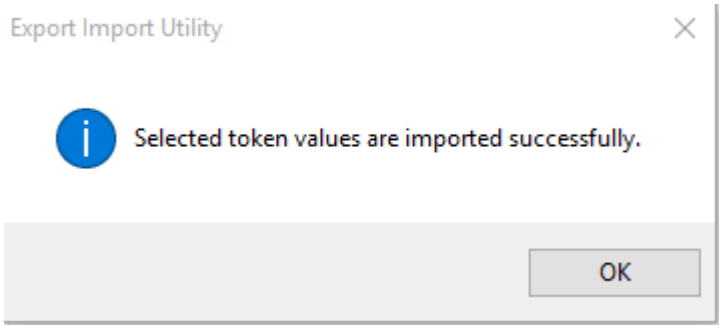


Figure 9

Import Flex Reports

1. Click **Reports** option, and then click the '**browse**'  button.
2. Locate applicable **Mimecast reports.etcrx** file, and then click the **Open** button.

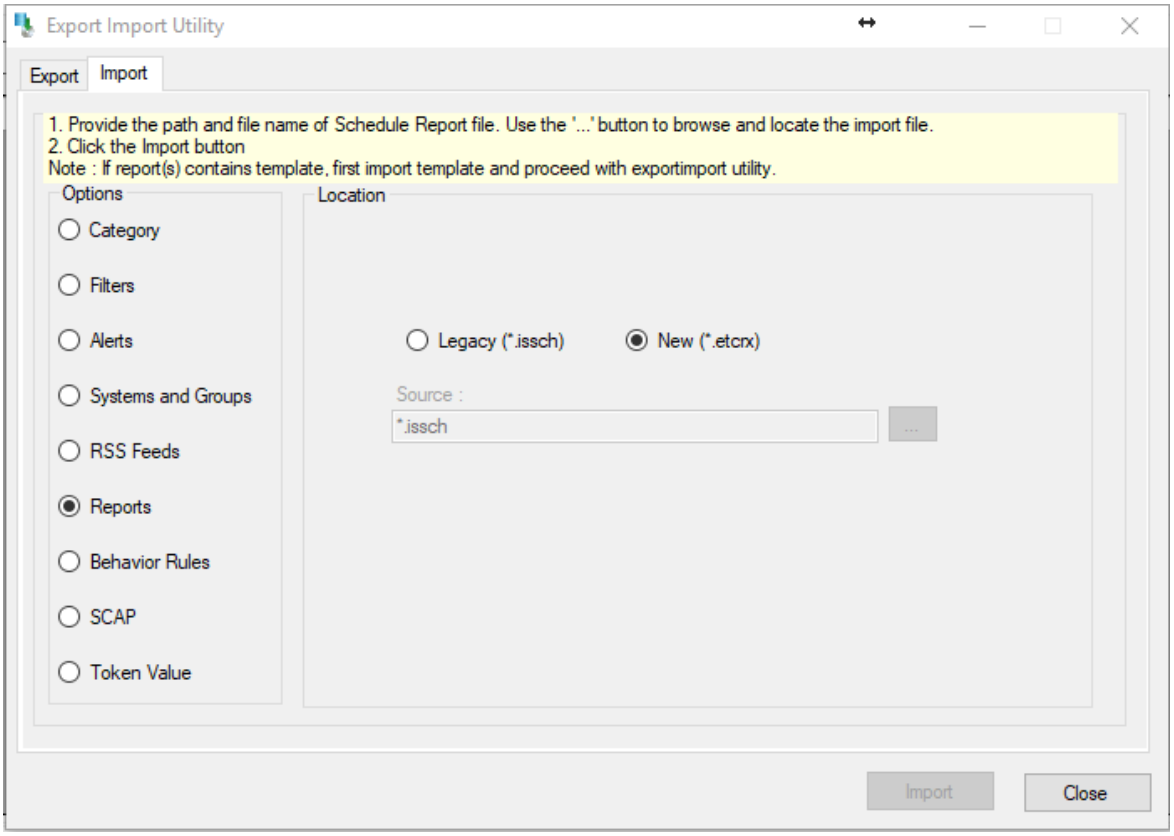


Figure 10

3. To import scheduled reports, click the **Import** button.

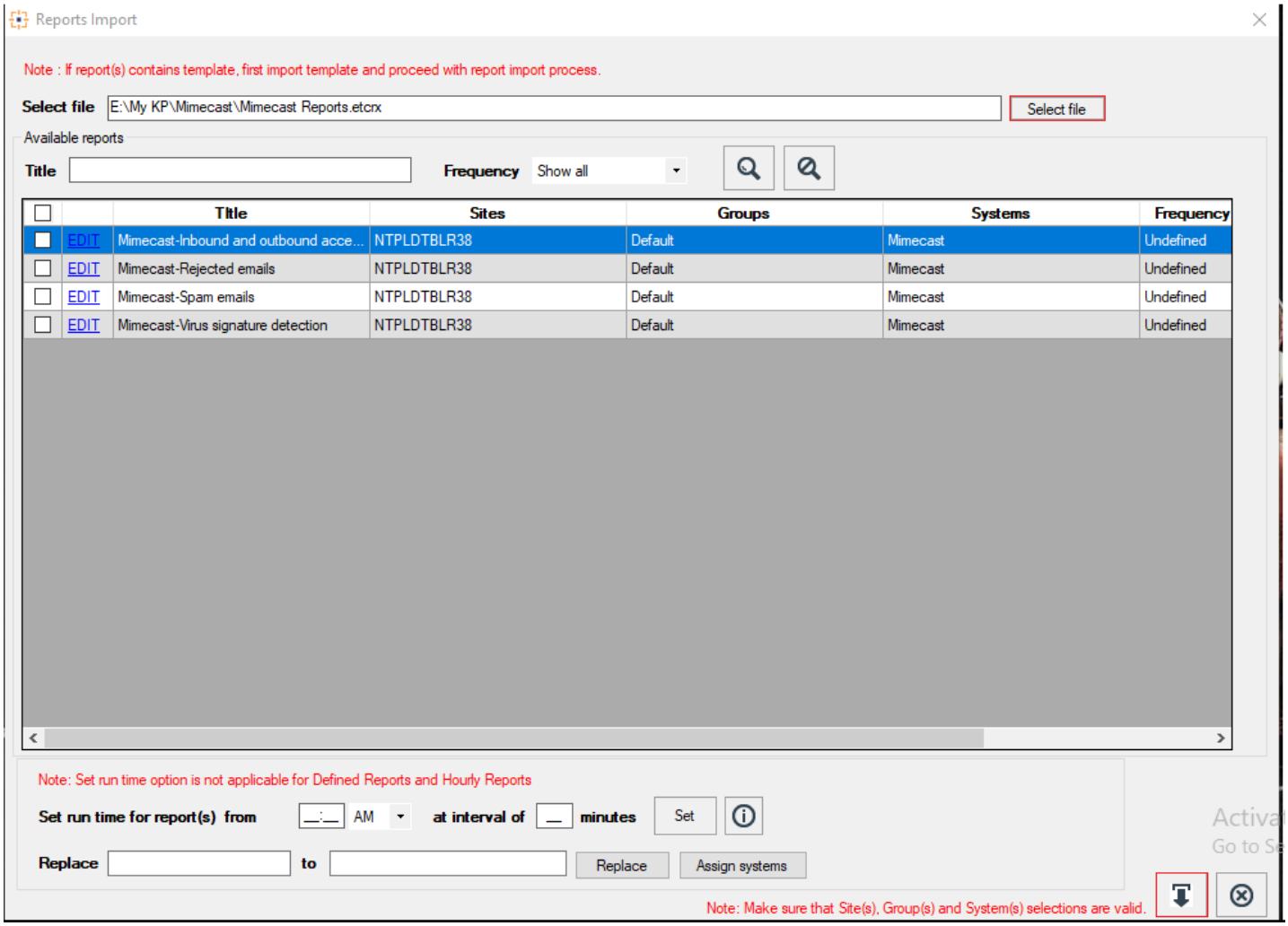


Figure 11

4. EventTracker displays success message.

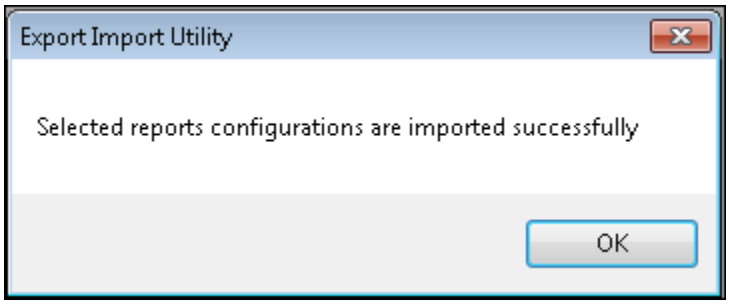


Figure 12

5. Click **OK**, and then click the **Close** button.

Verify Mimecast Secure Email Gateway Knowledge Pack

Verify Alerts

1. Logon to **EventTracker Enterprise**.
2. Click the **Admin** menu, and then click **Alerts**.
3. In the **Search** box, type '**Mimecast**, and then click the **Go** button.

   Alert Management page will display all the imported alerts.

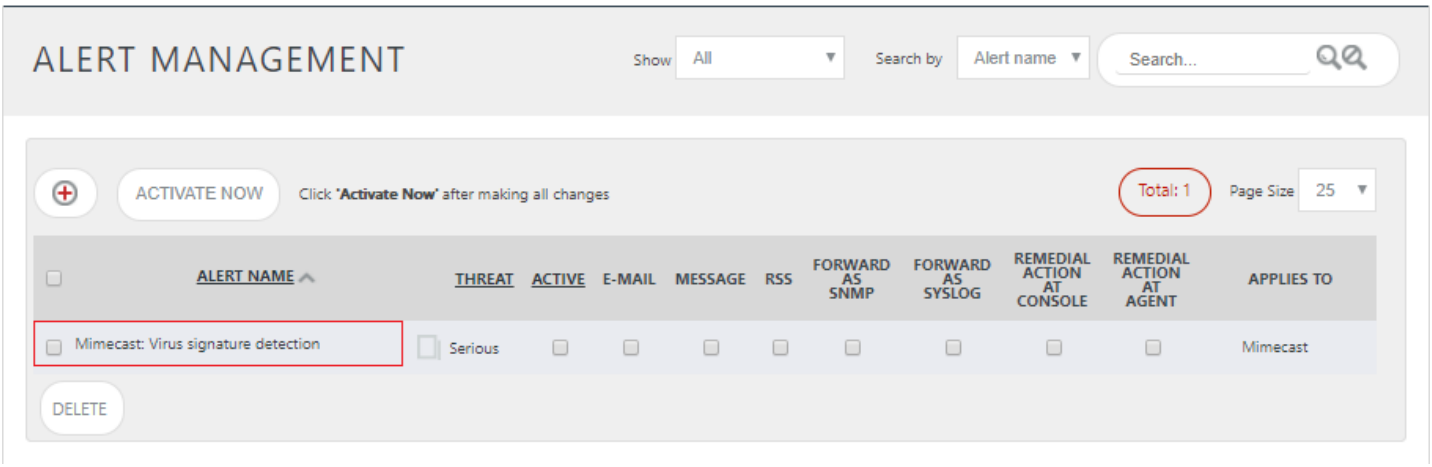


Figure 13

4. To activate the imported alerts, select the respective checkbox in the **Active** column.

   EventTracker displays message box.

Figure 14

5. Click **OK**, and then click the **Activate Now** button.

NOTE: Please specify appropriate **systems** in **alert configuration** for better performance.

Verify Knowledge Object

1. Click the **Admin** menu, and then click **Knowledge Objects**
2. Scroll down and select **Mimecast** in **Objects** pane.
   Imported Fortinet object details are shown.

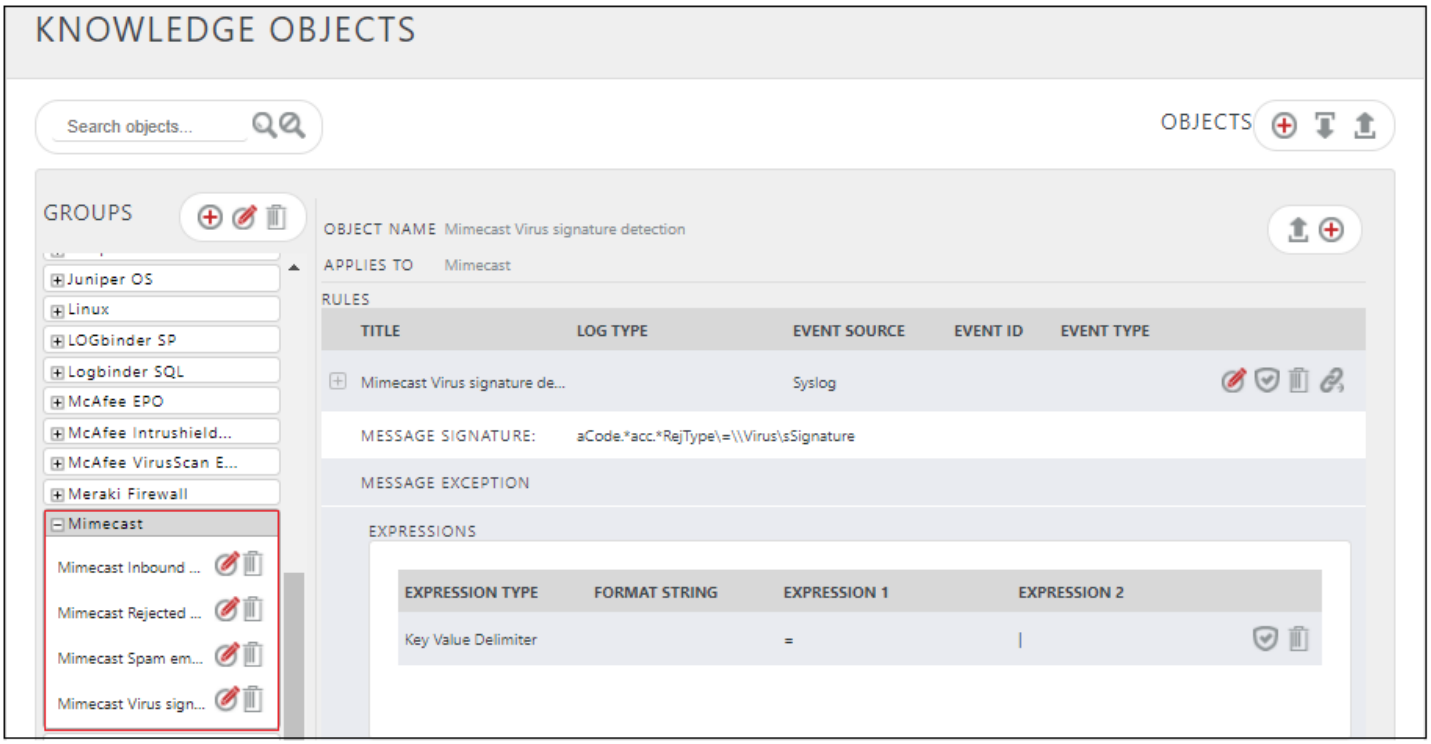


Figure 15

Parsing Rule

1. Logon to **EventTracker Enterprise** web interface.
2. Click the **Admin** menu, and then click **Parsing Rules.**
3. Click on **Mimecast** group option.

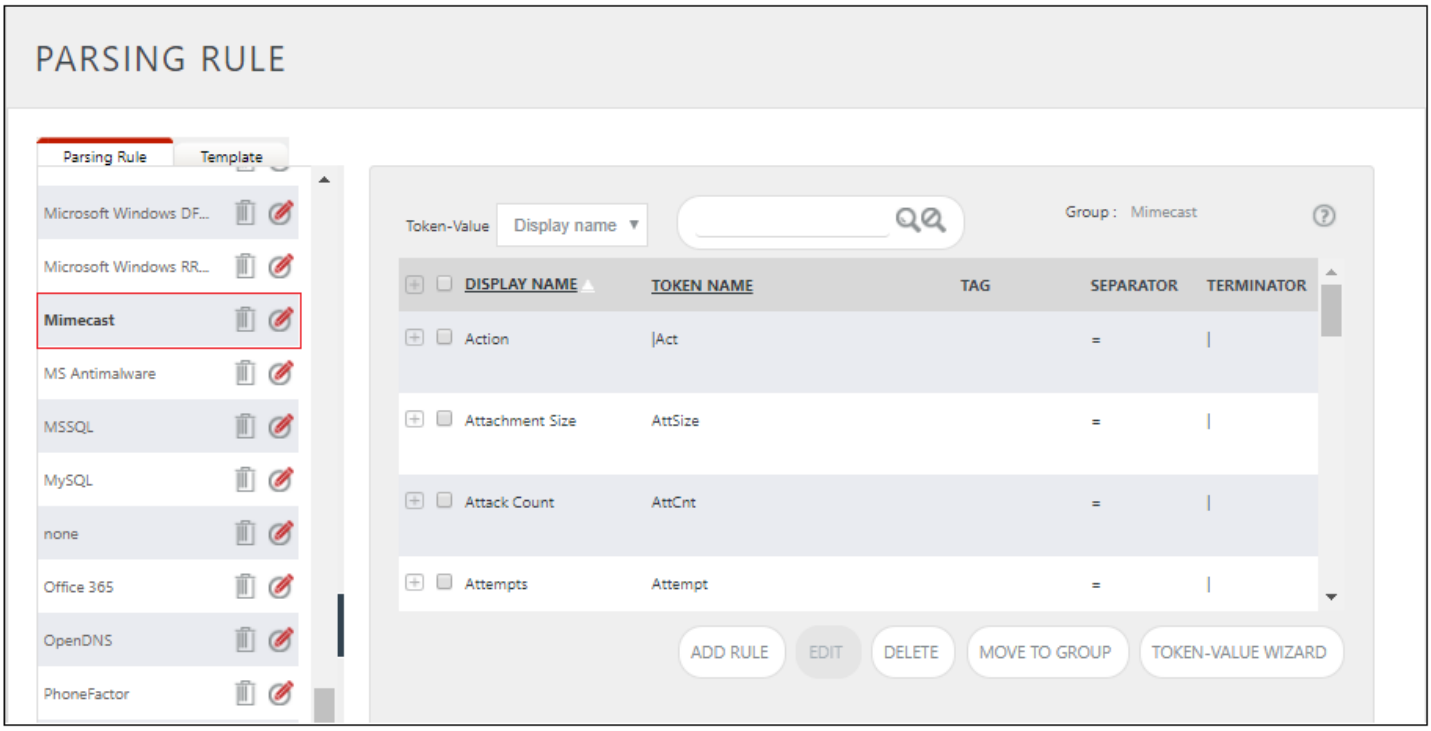


Figure 16

Verify Flex Reports

1. Logon to **EventTracker Enterprise**.
2. Click the **Reports** menu, and then **Configuration**.
3. Select **Defined** in report type.
4. In **Report Groups Tree** to view imported Scheduled Reports, scroll down and click **Mimecast** group folder.

   Scheduled Reports are displayed in the Reports configuration pane.

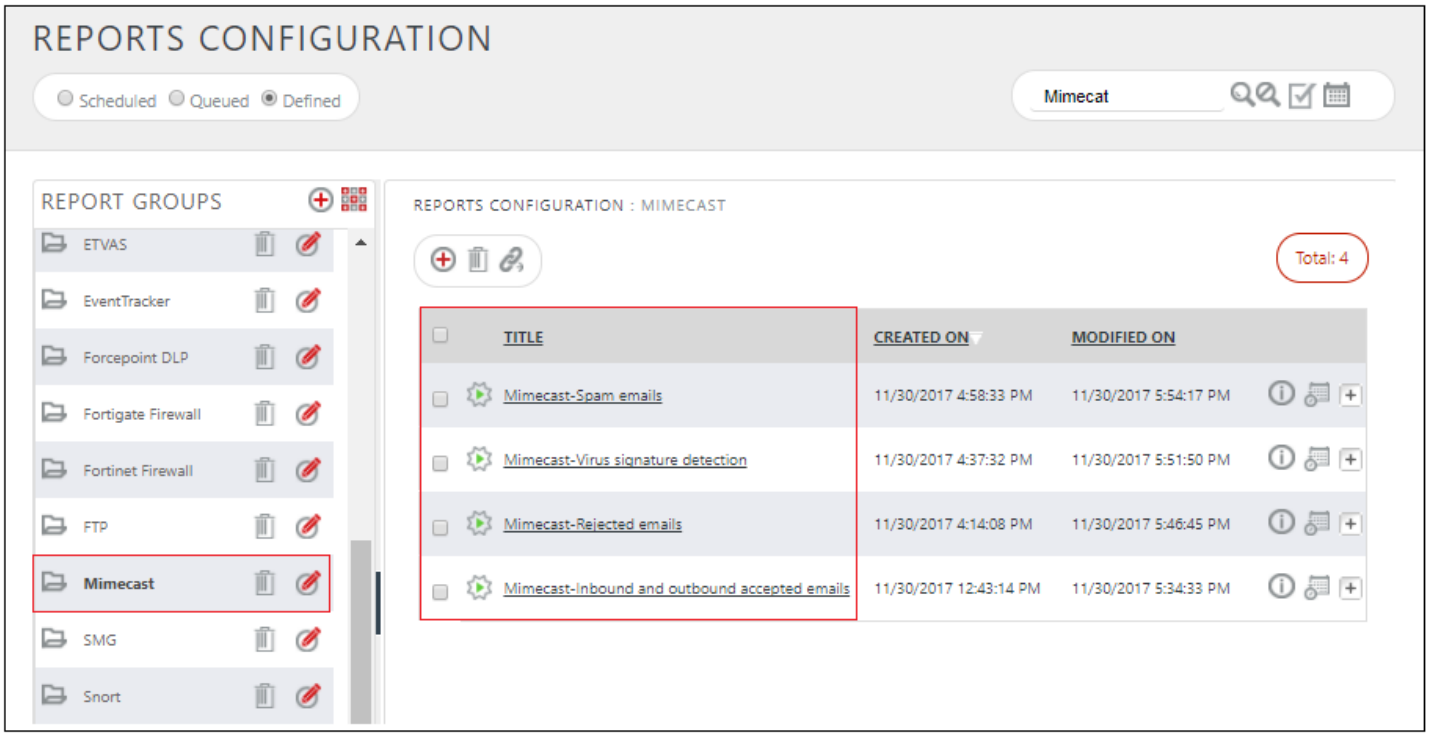


Figure 17

NOTE: Please specify appropriate **systems** in **report wizard** for better performance.

Create Dashboards in EventTracker

Schedule Reports

1. Open **EventTracker** in browser and logon.

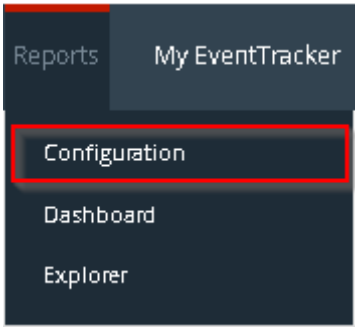


Figure 18

2. Navigate to **Reports>Configuration**.

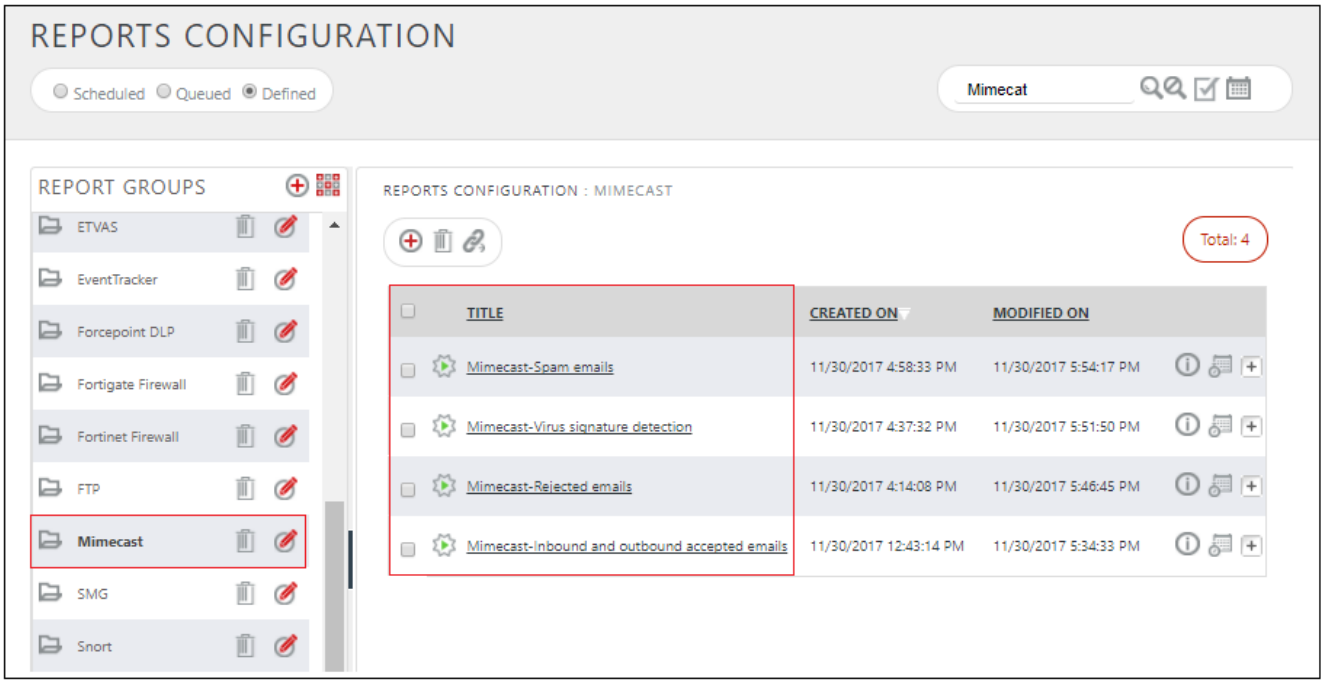


Figure 19

3. Select **Mimecast** in report groups. Check **defined** dialog box.
4. Click on '**schedule**'  to plan a report for later execution.

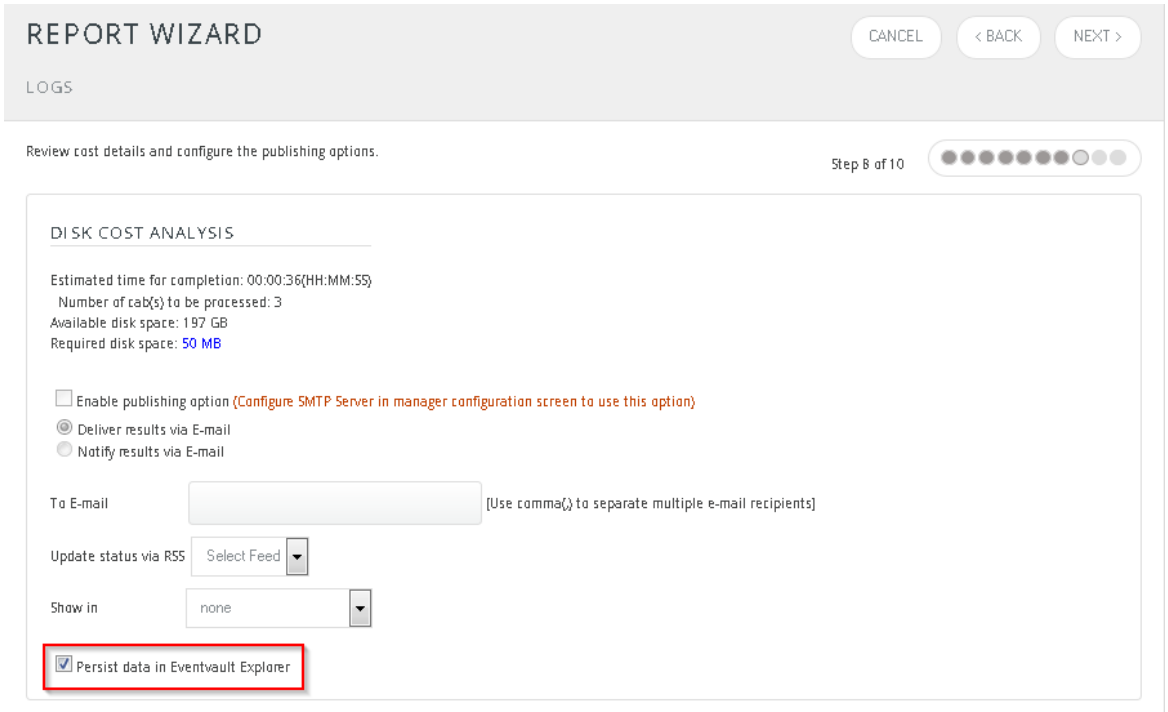


Figure 20

5. Choose appropriate time for report execution and in **Step 8** check **Persist data in Eventvault explorer** box.

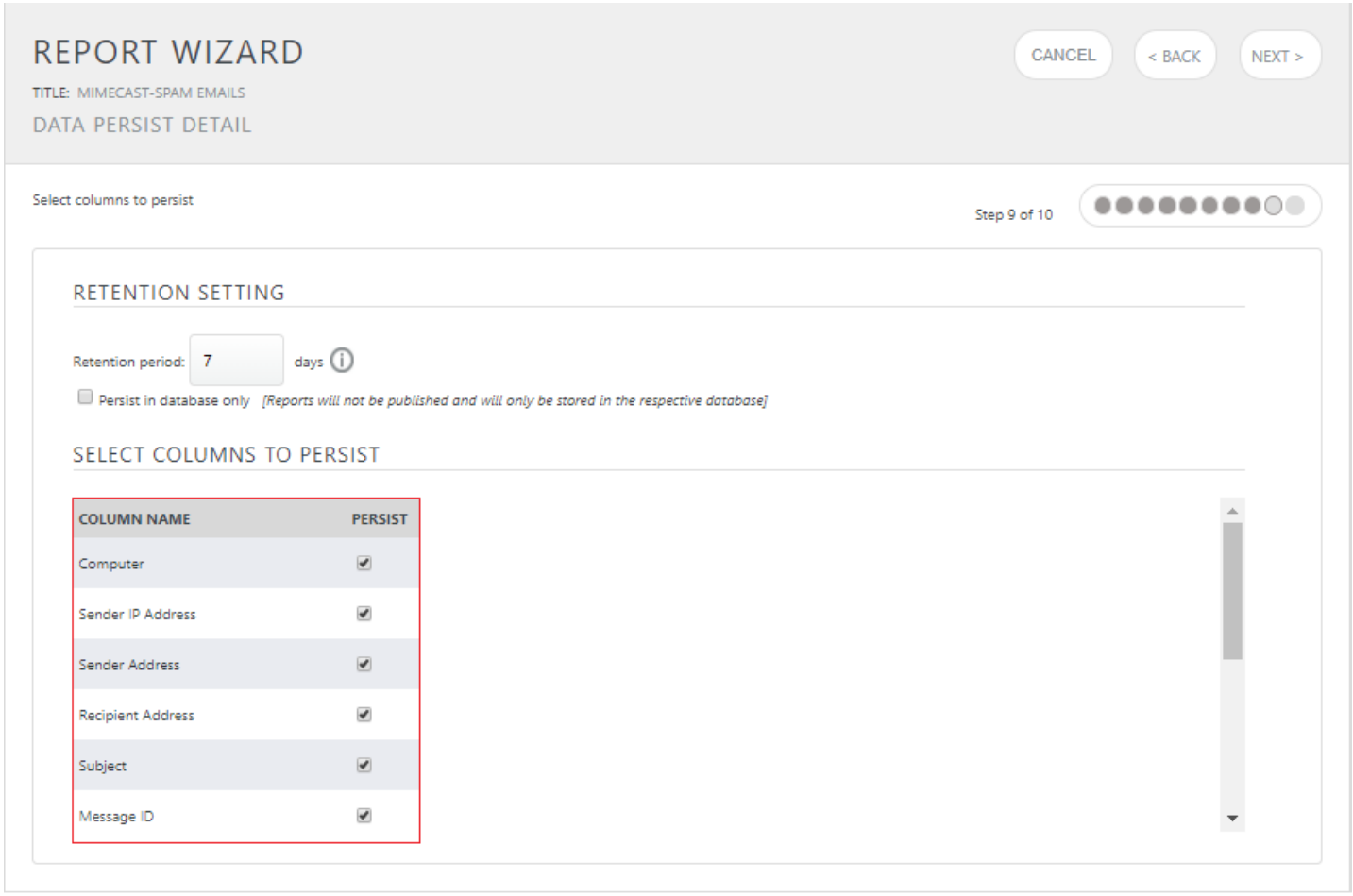


Figure 21

6. Check column names to persist using **PERSIST** checkboxes beside them. Choose suitable **Retention period**.
7. Proceed to next step and click **Schedule** button.
8. Wait for scheduled time or generate report manually.

Create Dashlets

1. **EventTracker 8** is required to configure flex dashboard.
2. Open **EventTracker** in browser and logon.

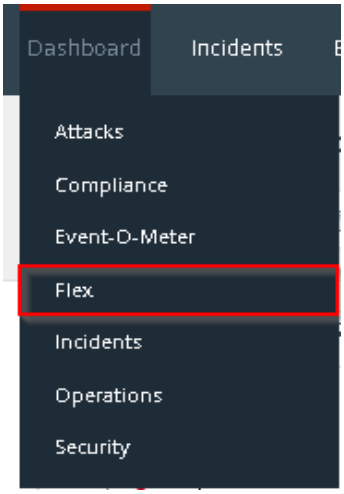


Figure 22

3. Navigate to **Dashboard>Flex**.
   Flex Dashboard pane is shown.

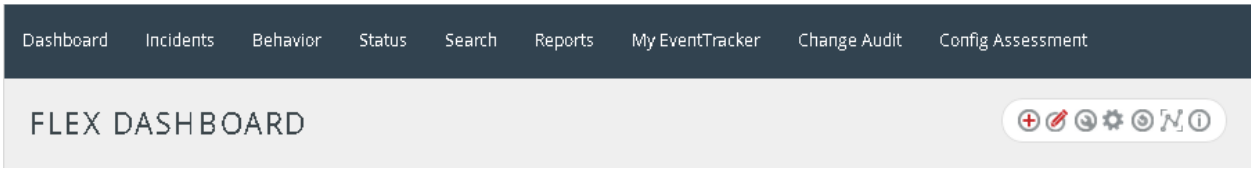


Figure 23

4. Click ⊕ to add a new dashboard.
   Flex Dashboard configuration pane is shown.

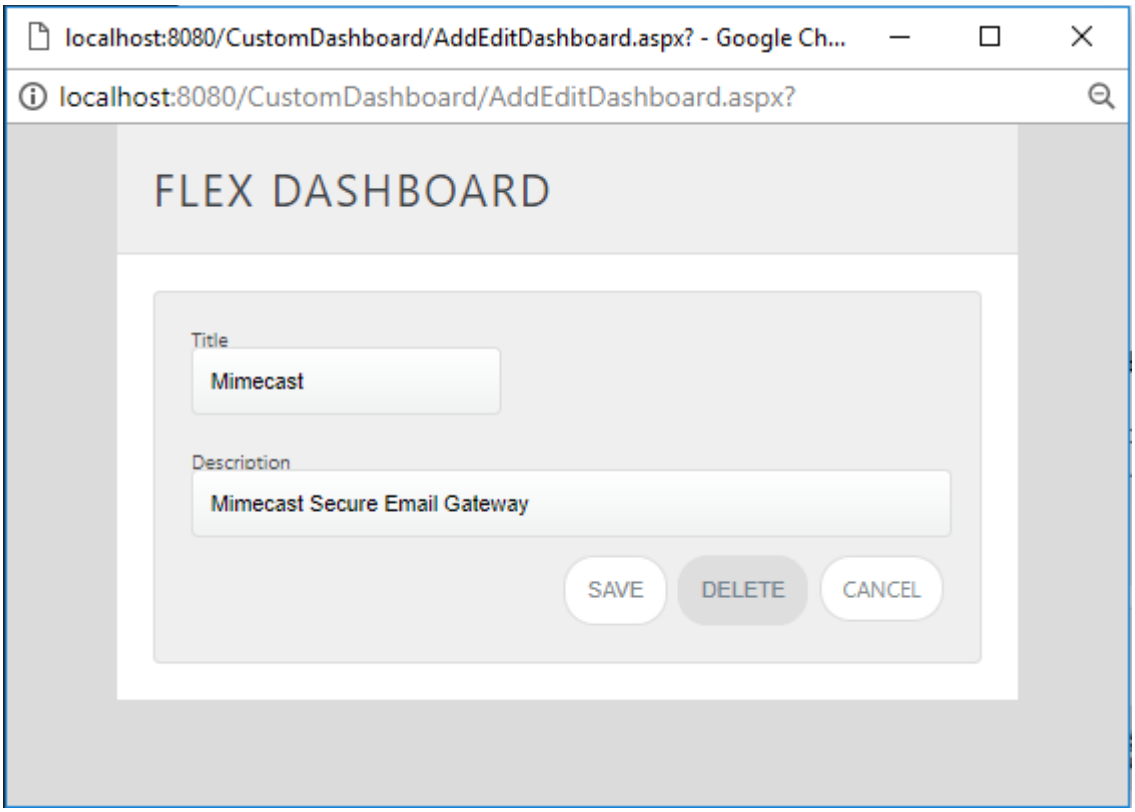


Figure 24

5. Fill fitting title and description and click **Save** button.
6. Click ⚙ to configure a new flex dashlet.
   Widget configuration pane is shown.

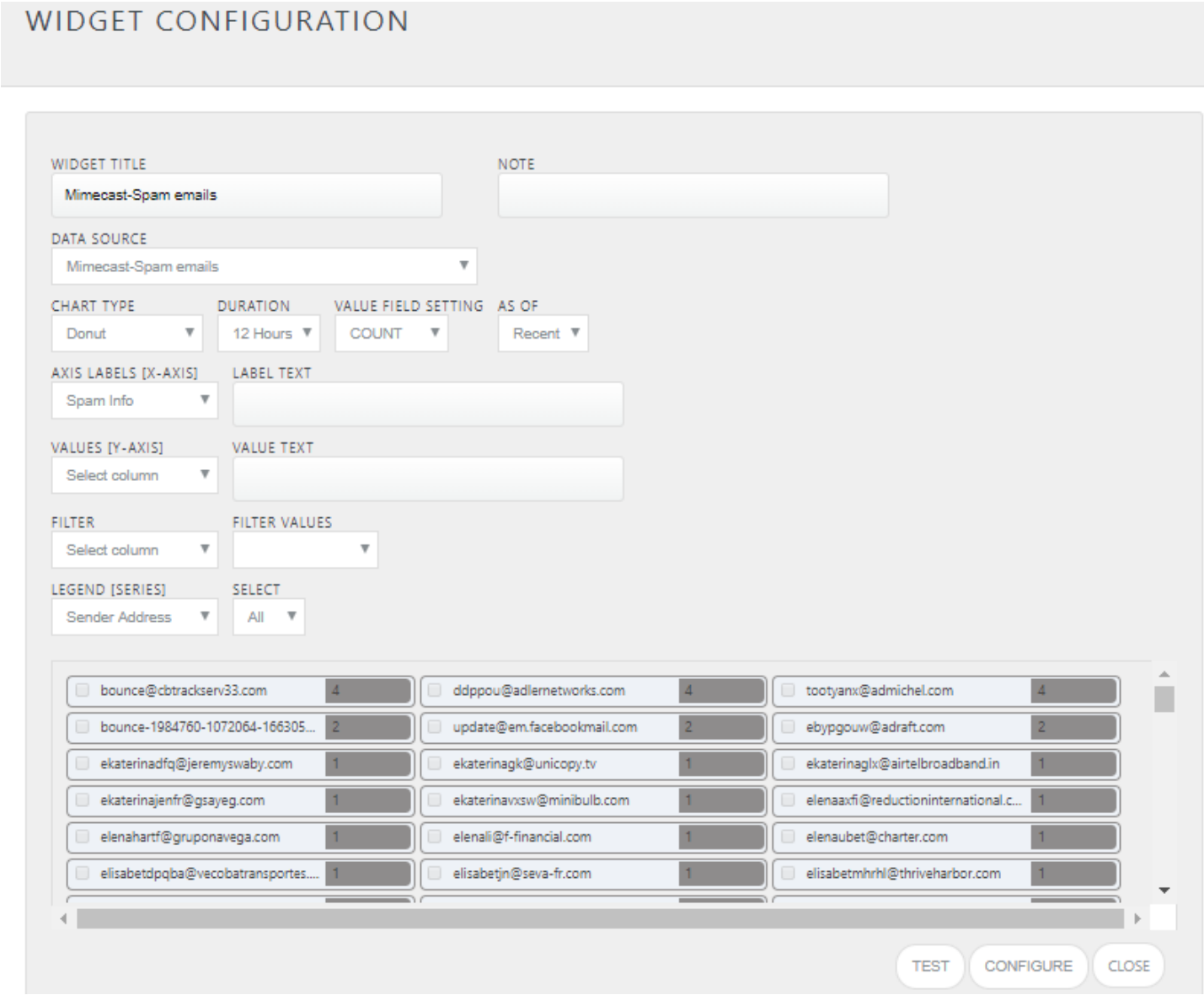


Figure 25

7. Locate earlier scheduled report in **Data Source** dropdown.
8. Select **Chart Type** from dropdown.
9. Select extent of data to be displayed in **Duration** dropdown.
10. Select computation type in **Value Field Setting** dropdown.
11. Select evaluation duration in **As Of** dropdown.
12. Select comparable values in **X Axis** with suitable label.
13. Select numeric values in **Y Axis** with suitable label.
14. Select comparable sequence in **Legend**.
15. Click **Test** button to evaluate.
    Evaluated chart is shown.

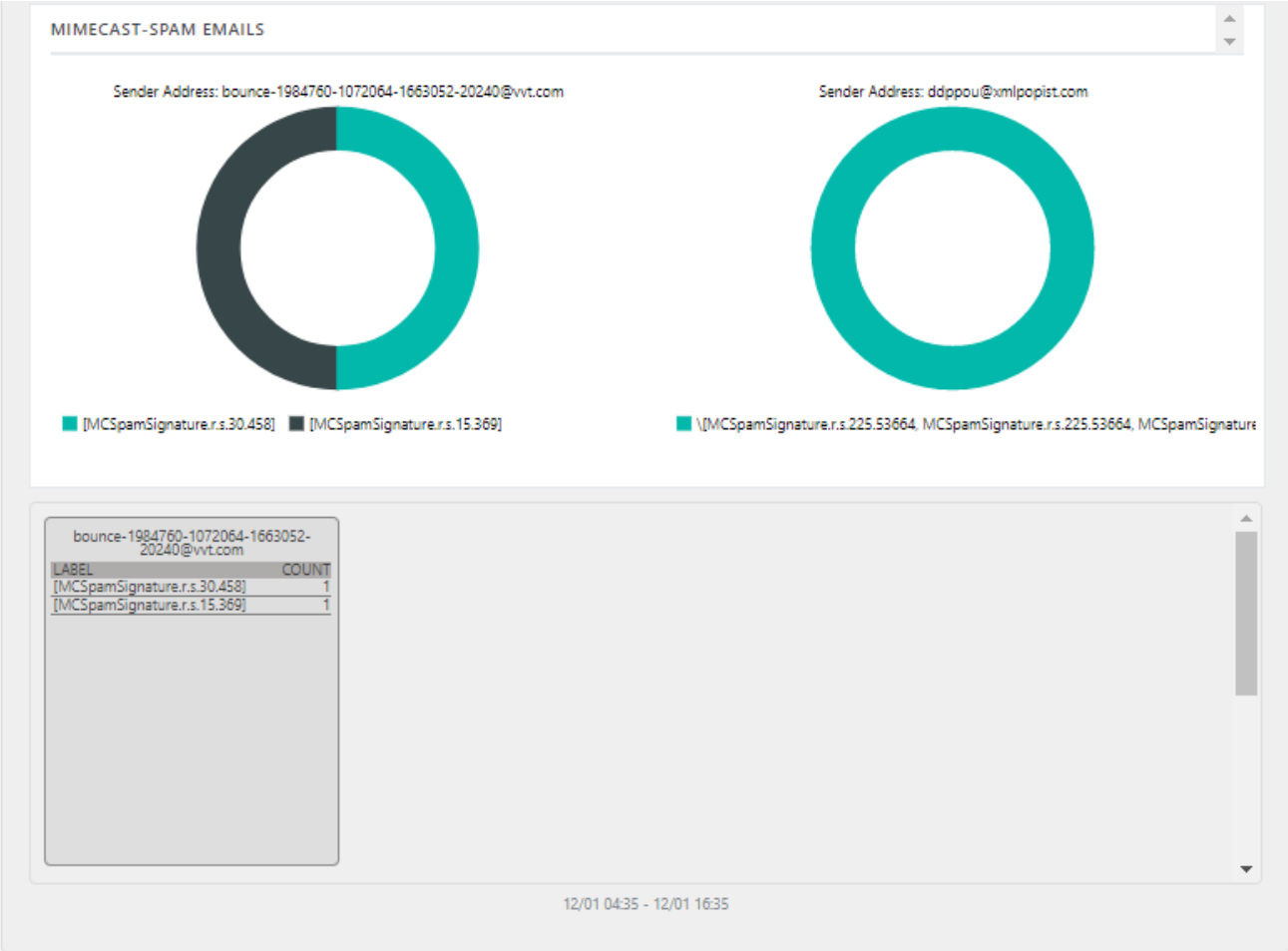


Figure 26

16. If satisfied, Click **Configure** button.

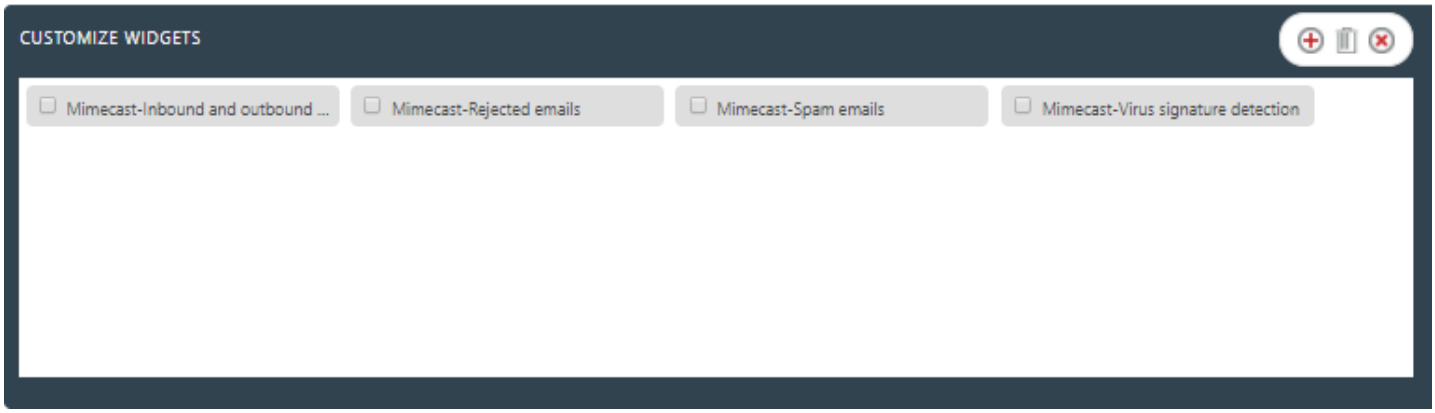
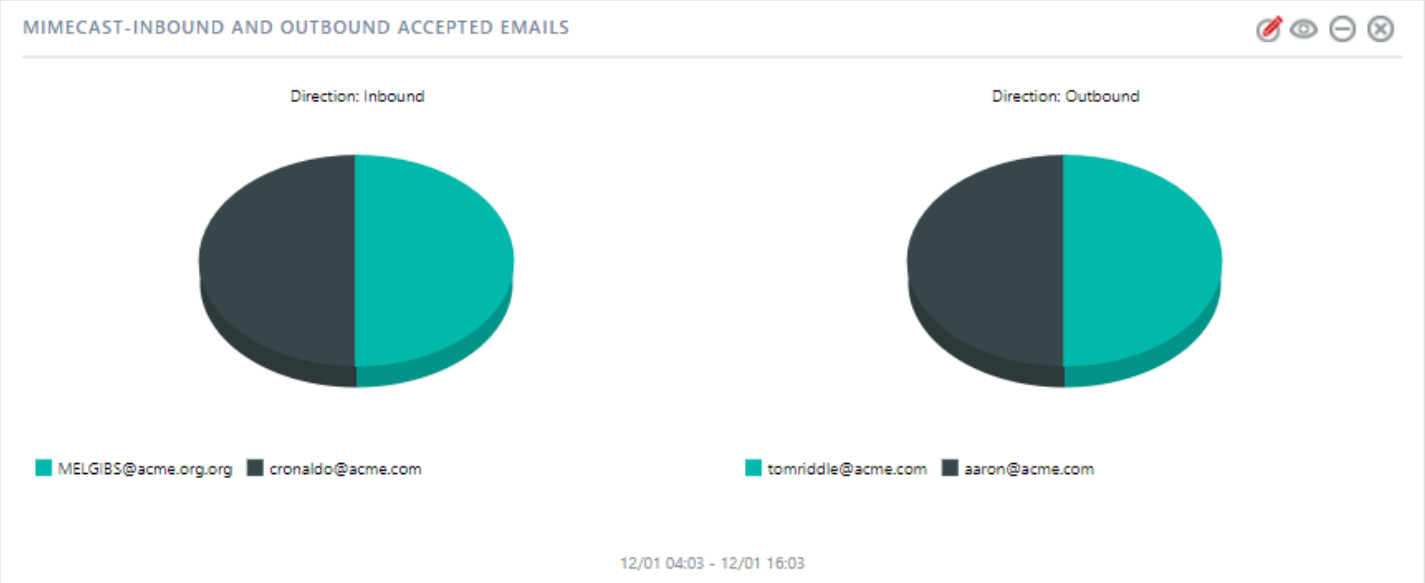


Figure 27

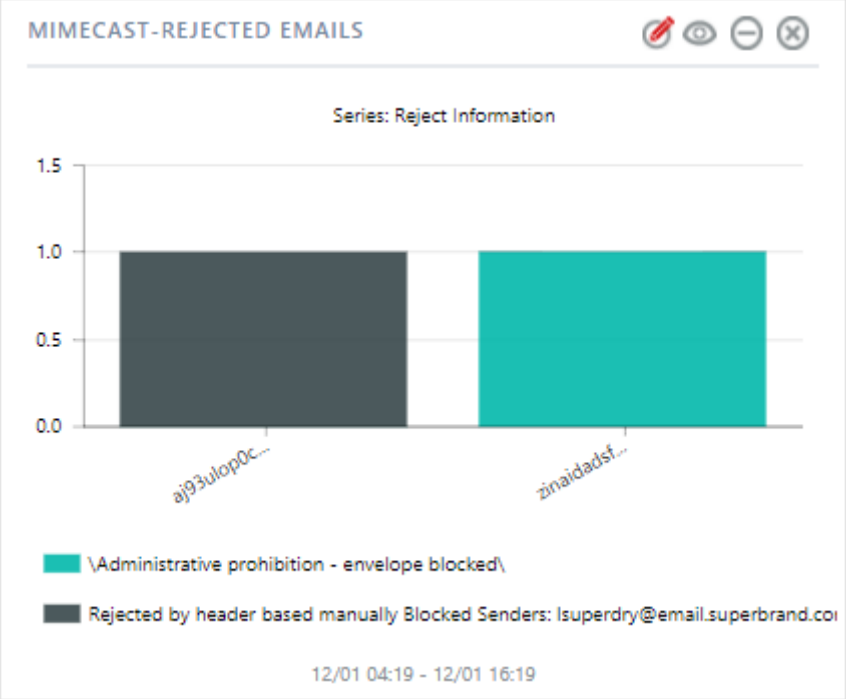
17. Click 'customize' to locate and choose created dashlet.
18. Click to add dashlet to earlier created dashboard.

Sample Dashboards

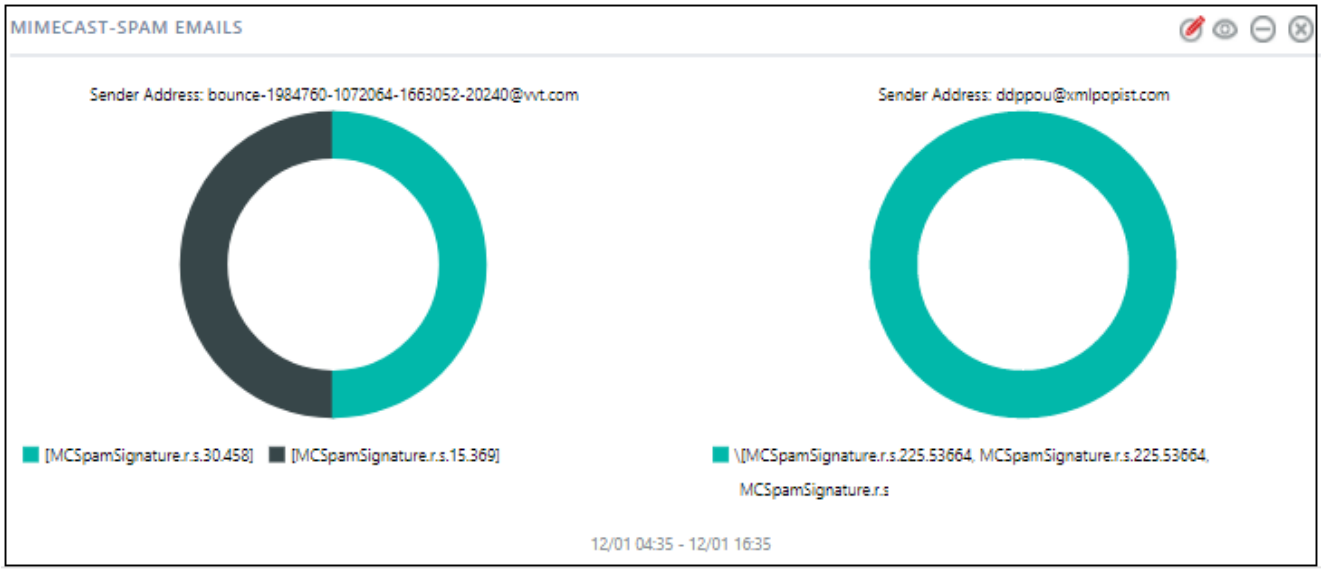
- **REPORT: Mimecast-Inbound and outbound accepted emails**
  WIDGET TITLE: Mimecast-Inbound and outbound accepted emails
  CHART TYPE: Pie
  AXIS LABELS [X-AXIS]: Sender Address
  LEGEND [SERIES]: Direction



- **REPORT: Mimecast-Rejected emails**
  WIDGET TITLE: Mimecast-Rejected emails
  CHART TYPE: Stacked Column
  AXIS LABELS [X-AXIS]: Sender Address
  LEGEND [SERIES]: Reject information



- **REPORT: Mimecast-Spam emails**
  WIDGET TITLE: Mimecast-Spam emails
  CHART TYPE: Donut
  AXIS LABELS [X-AXIS]: Spam Info
  LEGEND [SERIES]: Sender Address



- **REPORT: Mimecast-Virus signature detection**
  WIDGET TITLE: Mimecast-Virus signature detection
  CHART TYPE: Pie
  AXIS LABELS [X-AXIS]: Virus Details
  LEGEND [SERIES]: Sender Address