# Integrasi **Wazuh** dengan **YETI** (Your Everyday Threat Intelligence) - OpenCTI alter

1. Persiapkan VM untuk lingkungan YETI
   Kita pakai Ubuntu 24
2. Instalasi YETI dengan docker

```
git clone https://github.com/yeti-platform/yeti-docker
cd yeti-docker/prod
```

Kita turunkan spek arangodb ke versi 3.8.9 pada /yeti-docker/prod/docker-compose.yml

```
services:

  frontend:
    container_name: yeti-frontend
    build:
      dockerfile: ./frontend/Dockerfile
    image: yetiplatform/yeti-frontend:latest
    ports:
      - 80:80
    volumes:
      - ./frontend/nginx.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - api

  api:
    container_name: yeti-api
    image: yetiplatform/yeti:latest
    build:
      dockerfile: ./yeti/Dockerfile
    env_file:
      - ${YETI_DOCKER_ENVFILE}
    entrypoint:
      - /docker-entrypoint.sh
    command: ["webserver"]
    ports:
      - 8000:8000
    depends_on:
      - redis
      - arangodb
    volumes:
```

```
    - yeti-exports:/opt/yeti/exports

tasks:
  container_name: yeti-tasks
  image: yetiplatform/yeti:latest
  build:
    dockerfile: ./yeti/Dockerfile
  env_file:
    - ${YETI_DOCKER_ENVFILE}
  command: ['tasks']
  depends_on:
    - redis
    - arangodb
    - api
  volumes:
    - yeti-exports:/opt/yeti/exports

events-tasks:
  container_name: yeti-events-tasks
  image: yetiplatform/yeti:latest
  build:
    dockerfile: ./yeti/Dockerfile
  env_file:
    - ${YETI_DOCKER_ENVFILE}
  command: ['events-tasks']
  depends_on:
    - redis
    - arangodb
    - api
  volumes:
    - yeti-exports:/opt/yeti/exports

tasks-beat:
  container_name: yeti-tasks-beat
  image: yetiplatform/yeti:latest
  build:
    dockerfile: ./yeti/Dockerfile
  env_file:
    - ${YETI_DOCKER_ENVFILE}
  command: ['tasks-beat']
  depends_on:
    - redis

redis:
  container_name: yeti-redis
  image: redis:latest

arangodb:
  container_name: yeti-arangodb
  image: arangodb:3.8.9
  restart: unless-stopped
  environment:
    - ARANGO_ROOT_PASSWORD=cangcimen
  volumes:
    - yeti-db:/var/lib/arangodb3
```

```
  bloomcheck:
    container_name: bloomcheck
    image: yetiplatform/bloomcheck:dev
    entrypoint: ['/app/bloomcheck', '-serve', '/bloomfilters']
    volumes:
      - ${YETI_BLOOM_FILTERS_DIR}:/bloomfilters:ro

networks:
  default:
    name: yeti_network

volumes:
  yeti-exports:
    driver: local
  yeti-db:
    driver: local
```

generate YETI_AUTH_SECRET_KEY

user@yeti-docker/prod $/bin/bash ./init.sh

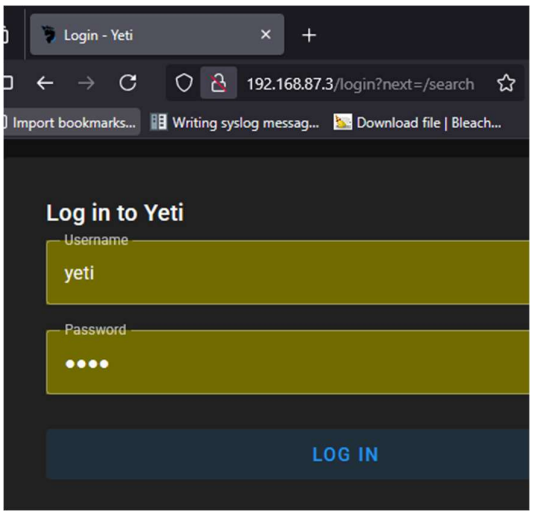
user@yeti-docker/prod $docker compose up -d

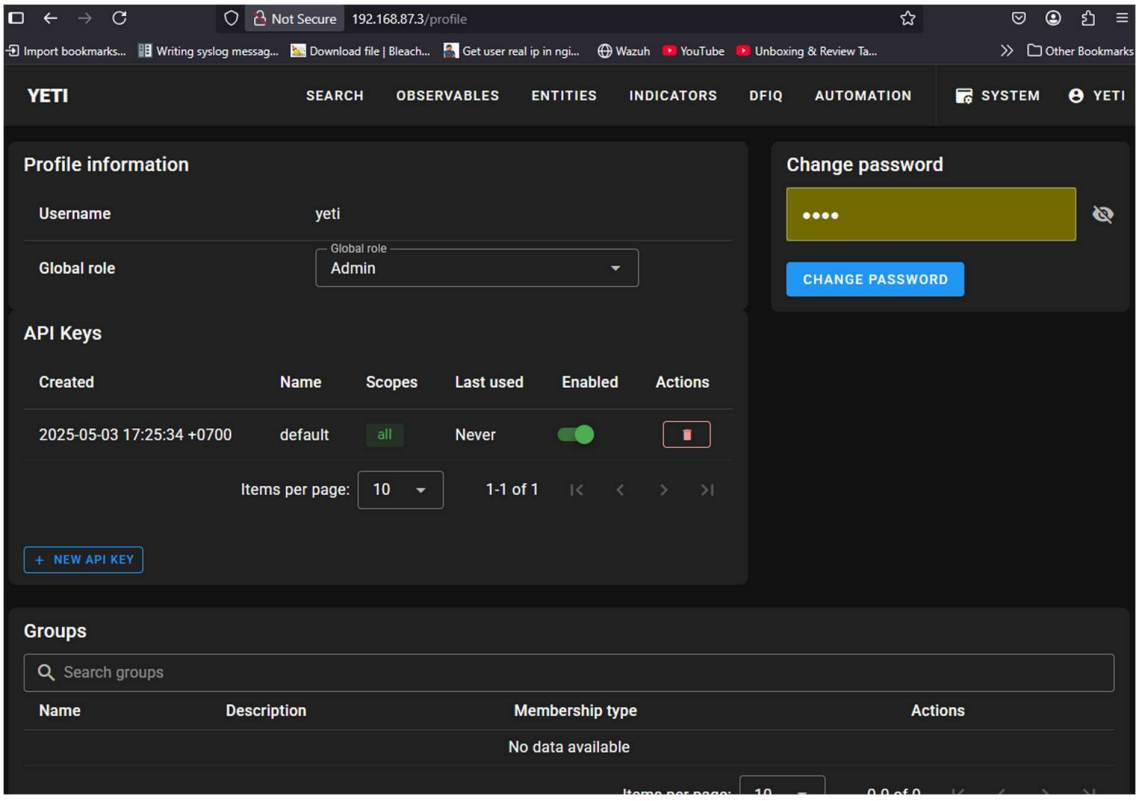
buat user

user@yeti-docker/prod$ docker compose run --rm api create-user USERNAME PASSWORD --admin

succesfully created! API key:
yeti:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiZGVmYXVsdCIsInN1YiI6InlldGkiLCJzY29wZXMiOlsiYWxsIl0sImNyZWF0ZWQiOiIyMDI1LTA1LTAzVDEwOjI1OjM0LjU2MTEzNVoiLCJleHAiOm51bGwsImxhc3RfdXNlZCI6bnVsbCwiZW5hYmxlZCI6dHJ1ZSwiZXhwaXJlZCI6ZmFsc2V9.vFe-_K53yWT6oDdL6QO8AbqwqTxZRfeH-QaAxFv-r7o

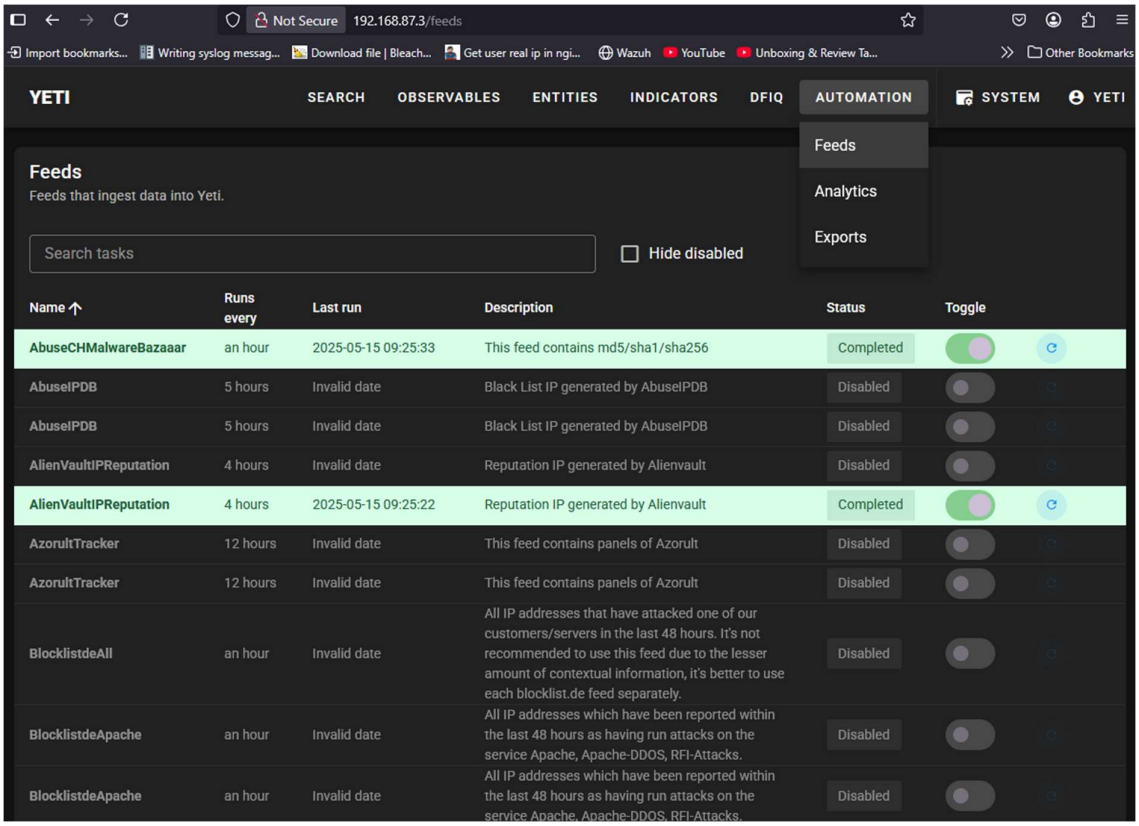
**cek keberhasilan instalasi**




Aktifkan Automation -> Feeds :

AbuseCHMalwareBazar & AlienVaultIPReputation




3. Proses integrasi

Buat Script Pemersatu kita 😅 :
Buat file /var/ossec/integrations/custom-yeti.py di sisi Wazuh Server

```
user@wazuh$sudo nano var/ossec/integrations/custom-yeti.py
```

```
#!/var/ossec/framework/python/bin/python3
import json
import os
import re
import sys
import requests
from requests.exceptions import Timeout
from socket import AF_UNIX, SOCK_DGRAM, socket

# Exit error codes
ERR_NO_REQUEST_MODULE = 1
ERR_BAD_ARGUMENTS = 2
```

```
ERR_BAD_MD5_SUM = 3
ERR_NO_RESPONSE_YETI = 4
ERR_SOCKET_OPERATION = 5
ERR_FILE_NOT_FOUND = 6
ERR_INVALID_JSON = 7

# Global vars
debug_enabled = True
timeout = 10
retries = 3
pwd = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
json_alert = {}

# Log and socket path
LOG_FILE = f'{pwd}/logs/integrations.log'
SOCKET_ADDR = f'{pwd}/queue/sockets/queue'

# Constants
ALERT_INDEX = 1
APIKEY_INDEX = 2
TIMEOUT_INDEX = 6
RETRIES_INDEX = 7
YETI_INSTANCE = 'http://192.168.56.1'

def debug(msg: str) -> None:
    """Log the message in the log file with the timestamp, if debug
flag
    is enabled."""
    if debug_enabled:
        print(msg)
        with open(LOG_FILE, 'a') as f:
            f.write(msg + '\n')

def main(args):
    global debug_enabled
    global timeout
    global retries
    try:
        # Read arguments
        bad_arguments: bool = False
        msg = ''
        if len(args) >= 4:
            debug_enabled = len(args) > 4 and args[4] == 'debug'
            if len(args) > TIMEOUT_INDEX:
                timeout = int(args[TIMEOUT_INDEX])
            if len(args) > RETRIES_INDEX:
                retries = int(args[RETRIES_INDEX])
        else:
            msg = '# Error: Wrong arguments\n'
            bad_arguments = True
```

```
        # Logging the call
        with open(LOG_FILE, 'a') as f:
            f.write(msg)

        if bad_arguments:
            debug('# Error: Exiting, bad arguments. Inputted: %s' %
args)
            sys.exit(ERR_BAD_ARGUMENTS)

        # Read args
        apikey: str = args[APIKEY_INDEX]

        # Obtain the access token
        access_token = getAccessToken(apikey)

        # Core function
        process_args(args, access_token)


    except Exception as e:
        debug(str(e))
        raise

def getAccessToken(apikey):
    """Exchange API key for a JWT access token."""

    url = f"{YETI_INSTANCE}/api/v2/auth/api-token"
    headers = {"x-yeti-apikey": apikey}
    try:
        response = requests.post(url, headers=headers)
        response.raise_for_status()
        access_token = response.json().get("access_token")
        if not access_token:
            raise ValueError("Access token missing in the
response.")
        return access_token
    except requests.exceptions.RequestException as e:
        debug(f"Error obtaining access token from API: {e}")
        sys.exit(1)

def process_args(args, access_token: str) -> None:
    """This is the core function, creates a message with all valid
fields
    and overwrite or add with the optional fields."""
    debug('# Running Yeti script')

    # Read args
    alert_file_location: str = args[ALERT_INDEX]
```

```
    # Load alert. Parse JSON object.
    json_alert = get_json_alert(alert_file_location)
    debug(f"# Opening alert file at '{alert_file_location}' with
'{json_alert}'")

    # Determine the type of alert and process accordingly
    if 'data' in json_alert and ('sshd' in json_alert or 'srcip' in
json_alert['data']):
        debug('# Detected an SSH-related alert')
        msg: any = request_ssh_info(json_alert, access_token)

    elif 'syscheck' in json_alert or 'md5_after' in
json_alert['syscheck']:
        debug('# Detected a file integrity alert (MD5 check)')
        msg: any = request_md5_info(json_alert, access_token)

    else:
        debug('# Alert does not match known types (SSH or MD5).
Skipping processing.')
        return None

    # If a valid message is generated, send it
    if msg:
        send_msg(msg, json_alert['agent'])
    else:
        debug('# No valid message generated. Skipping sending.')

def get_json_alert(file_location: str) -> any:
    """Read JSON alert object from file."""
    try:
        with open(file_location) as alert_file:
            return json.load(alert_file)
    except FileNotFoundError:
        debug("# JSON file for alert %s doesn't exist" %
file_location)
        sys.exit(ERR_FILE_NOT_FOUND)
    except json.decoder.JSONDecodeError as e:
        debug('Failed getting JSON alert. Error: %s' % e)
        sys.exit(ERR_INVALID_JSON)

def request_ssh_info(alert: any, access_token: str):
    """Generate the JSON object with the message to be send."""
    alert_output = {'yeti': {}, 'integration': 'yeti'}

    # Extract source IP
    src_ip = alert['data']['srcip']

    # Inline validation of the source IP
    if not isinstance(src_ip, str):
        debug(f"# Invalid src_ip: '{src_ip}' is not a string")
```

```
        return None

    octets = src_ip.split('.')
    if len(octets) != 4 or not all(octet.isdigit() for octet in
octets):
        debug(f"# Invalid src_ip format: '{src_ip}'")
        return None

    octets = list(map(int, octets))
    if (
        any(octet < 0 or octet > 255 for octet in octets) or  #
Octet range validation
        octets[0] in [10, 127] or  # Exclude private (10.x.x.x) and
loopback (127.x.x.x)
        (octets[0] == 192 and octets[1] == 168) or  # Exclude
private (192.168.x.x)
        (octets[0] == 172 and 16 <= octets[1] <= 31) or  # Exclude
private (172.16.x.x to 172.31.x.x)
        octets[0] >= 240  # Exclude reserved and invalid ranges
(240.x.x.x and above)
    ):
        debug(f"# Invalid src_ip: '{src_ip}' is private, reserved,
or out of range")
        return None

    # Request info using Yeti API
    yeti_response_data = request_info_from_api(alert_output,
access_token, src_ip)

    if not yeti_response_data:
        debug("No data returned from the Yeti API.")
        return None

    alert_output['yeti']['source'] = {
        'alert_id': alert['id'],
        'src_ip': alert['data']['srcip'],
        'src_port': alert['data']['srcport'],
        'dst_user': alert['data']['dstuser'],
    }

    # Check if Yeti has any info about the source IP
    """Filter YETI results for entries with source
'AlienVaultIPReputation'."""
    if not yeti_response_data:
        debug("No data returned from the YETI API.")
        return None

    observables = yeti_response_data.get('observables', [])
    for observable in observables:
        if isinstance(observable, dict):  # Ensure observable is a
```

```
dictionary
            for context_entry in observable.get("context", []):
                if context_entry.get("source") ==
"AlienVaultIPReputation":
                    observable_value = observable.get("value")
                    if observable_value == src_ip:
                        alert_output['yeti'].update(
                            {
                            'info': {
                                'country_code':
context_entry.get("country"),
                                'threat':
context_entry.get("threat"),
                                'reliability':
context_entry.get("reliability"),
                                'risk': context_entry.get("risk"),
                                'source': "AlienVaultIPReputation",
                                }
                            }
                        )
                        return alert_output
        else:
            debug(f"Invalid observable format: {observable}")

    debug(f"No matching IP address '{src_ip}' found in YETI API for
source 'AlienVaultIPReputation'.")
    return None

def request_md5_info(alert: any, access_token: str):
    """Generate the JSON object with the message to be send."""
    alert_output = {'yeti': {}, 'integration': 'yeti'}

    # Extract MD5 hash
    md5_hash = alert['syscheck']['md5_after']

    # Validate MD5 hash
    if not isinstance(alert['syscheck']['md5_after'], str) or
len(re.findall(r'\b([a-f\d]{32}|[A-F\d]{32})\b',
alert['syscheck']['md5_after'])) != 1:
        debug(f"# Invalid md5_after value:
'{alert['syscheck']['md5_after']}'")
        return None

    # Request info using Yeti API
    yeti_response_data = request_info_from_api(alert_output,
access_token, md5_hash)

    if not yeti_response_data:
        debug("No data returned from the Yeti API.")
        return None
```

```
    alert_output['yeti']['source'] = {
        'alert_id': alert['id'],
        'file': alert['syscheck']['path'],
        'md5': alert['syscheck']['md5_after'],
        'sha1': alert['syscheck']['sha1_after'],
    }

    # Check if Yeti has any info about the hash
    """Filter YETI results for entries with source
'AbuseCHMalwareBazaaar'."""
    if not yeti_response_data:
        debug("No data returned from the YETI API.")
        return None

    observables = yeti_response_data.get('observables', [])
    for observable in observables:
        if isinstance(observable, dict):  # Ensure observable is a
dictionary
            for context_entry in observable.get("context", []):
                if context_entry.get("source") ==
"AbuseCHMalwareBazaaar":
                    observable_value = observable.get("value")
                    if observable_value == md5_hash:
                        alert_output['yeti'].update(
                            {
                            'info': {
                                'country_code':
context_entry.get("country"),
                                'threat':
context_entry.get("threat"),
                                'reliability':
context_entry.get("reliability"),
                                'risk': context_entry.get("risk"),
                                'source': "AbuseCHMalwareBazaaar",
                                }
                            }
                        )
                        return alert_output
        else:
            debug(f"Invalid observable format: {observable}")

    debug(f"No matching IP address '{md5_hash}' found in YETI API
for source 'AbuseCHMalwareBazaaar'.")
    return None


def request_info_from_api(alert_output, access_token, src_ip):
    """Request information from Yeti API."""
    for attempt in range(retries + 1):
```

```
        try:
            yeti_response_data = query_api(src_ip, access_token)
            return yeti_response_data
        except Timeout:
            debug('# Error: Request timed out. Remaining retries:
%s' % (retries - attempt))
            continue
        except Exception as e:
            debug(str(e))
            sys.exit(ERR_NO_RESPONSE_YETI)

    debug('# Error: Request timed out and maximum number of retries
was exceeded')
    alert_output['yeti']['error'] = 408
    alert_output['yeti']['description'] = 'Error: API request timed
out'
    send_msg(alert_output)
    sys.exit(ERR_NO_RESPONSE_YETI)

def query_api(src_ip, access_token: str) -> any:
    """Query the API for observables."""
    headers = {
        "Authorization": f"Bearer {access_token}",
        "Content-Type": "application/json"
    }
    data = json.dumps({
        "page": 0,
        "count": 25,
        "query": {"value": src_ip},
        "sorting": [["created", False]]
    })
    debug('# Querying Yeti API')
    response = requests.post(
        f'{YETI_INSTANCE}/api/v2/observables/search',
headers=headers, data=data, timeout=timeout
    )
    if response.status_code == 200:
        return response.json()
    else:
        handle_api_error(response.status_code)

def handle_api_error(status_code):
    """Handle errors from the Yeti API."""
    alert_output = {}
    alert_output['yeti'] = {}
    alert_output['integration'] = 'yeti'

    if status_code == 401:
        alert_output['yeti']['error'] = status_code
        alert_output['yeti']['description'] = 'Error: Unauthorized.
```

```
Check your API key.'
        send_msg(alert_output)
        raise Exception('# Error: Yeti credentials, required
privileges error')
    elif status_code == 404:
        alert_output['yeti']['error'] = status_code
        alert_output['yeti']['description'] = 'Error: Resource not
found.'
    elif status_code == 500:
        alert_output['yeti']['error'] = status_code
        alert_output['yeti']['description'] = 'Error: Internal
server error.'
    else:
        alert_output['yeti']['error'] = status_code
        alert_output['yeti']['description'] = 'Error: API request
failed.'

    send_msg(alert_output)
    raise Exception(f'# Error: Yeti API request failed with status
code {status_code}')

def send_msg(msg: any, agent: any = None) -> None:
    if not agent or agent['id'] == '000':
        string = '1:yeti:{0}'.format(json.dumps(msg))
    else:
        location = '[{0}] ({1}) {2}'.format(agent['id'],
agent['name'], agent['ip'] if 'ip' in agent else 'any')
        location = location.replace('|', '||').replace(':', '|:')
        string = '1:{0}->yeti:{1}'.format(location, json.dumps(msg))

    debug('# Request result from Yeti server: %s' % string)
    try:
        sock = socket(AF_UNIX, SOCK_DGRAM)
        sock.connect(SOCKET_ADDR)
        sock.send(string.encode())
        sock.close()
    except FileNotFoundError:
        debug('# Error: Unable to open socket connection at %s' %
SOCKET_ADDR)
        sys.exit(ERR_SOCKET_OPERATION)

if __name__ == '__main__':
    main(sys.argv)
```

**Registrasikan ke Wazuh-manager**

Edit file /var/ossec/etc/ossec.conf, tambahkan ke dalam <ossec_config>:

```
<integration>
  <name>custom-yeti.py</name>
  <api_key><YETI_API_KEY></api_key>
  <group>syscheck,sshd,ids</group>
  <alert_format>json</alert_format>
</integration>
```

Ganti **<YETI_API_KEY>** dengan **API saat create user** menjadi seperti ini:

```
<integration>
  <name>custom-yeti.py</name>

<api_key>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiZGVmYXVsdCIsInN1Y
iI6InlldGkiLCJzY29wZXMiOlsiYWxsIl0sImNyZWF0ZWQiOiIyMDI1LTA1LTAzVDEwOjI1OjM0
LjU2MTEzNVoiLCJleHAiOm51bGwsImxhc3RfdXNlZCI6bnVsbCwiZW5hYmxlZCI6dHJ1ZSwiZXh
waXJlZCI6ZmFsc2V9.vFe-_K53yWT6oDdL6QO8AbqwqTxZRfeH-QaAxFv-r7o</api_key>
  <group>syscheck,sshd,ids,syslog</group>
  <alert_format>json</alert_format>
</integration>
```

Definisikan Alert

```
user@wazuh$sudo nano /var/ossec/etc/rules/yeti_rules.xml
```

```
<group name="yeti,">
  <rule id="100500" level="0">
    <decoded_as>json</decoded_as>
    <field name="integration">yeti</field>
    <description>yeti integration messages.</description>
    <options>no_full_log</options>
  </rule>
  <rule id="100501" level="12">
    <if_sid>100500</if_sid>
    <field name="yeti.info.source">AbuseCHMalwareBazaaar</field>
    <description>"Yeti Alert - " $(yeti.info.source) detected this
file: $(yeti.source.file)</description>
    <group>pci_dss_10.6.1,pci_dss_11.4,gdpr_IV_35.7.d,</group>
    <options>no_full_log</options>
    <mitre>
      <id>T1203</id>
    </mitre>
  </rule>
  <rule id="100502" level="12">
    <if_sid>100500</if_sid>
```

```
    <field name="yeti.info.source">AlienVaultIPReputation</field>
    <description>"Yeti Alert - " $(yeti.info.source) detected IP
address: $(yeti.source.src_ip)</description>
    <group>pci_dss_10.2.4,pci_dss_10.2.5,</group>
    <options>no_full_log</options>
  </rule>
</group>
```

```
Restart wazuh-manager
```

```
user@wazuh$sudo service wazuh-manager restart
```

4. Konfigurasi endpoin untuk Lokasi file dan direktori yang dipantau

   **Malware (FIM) use case**

   nano /var/ossec/etc/ossec.conf di dalam <syscheck>:

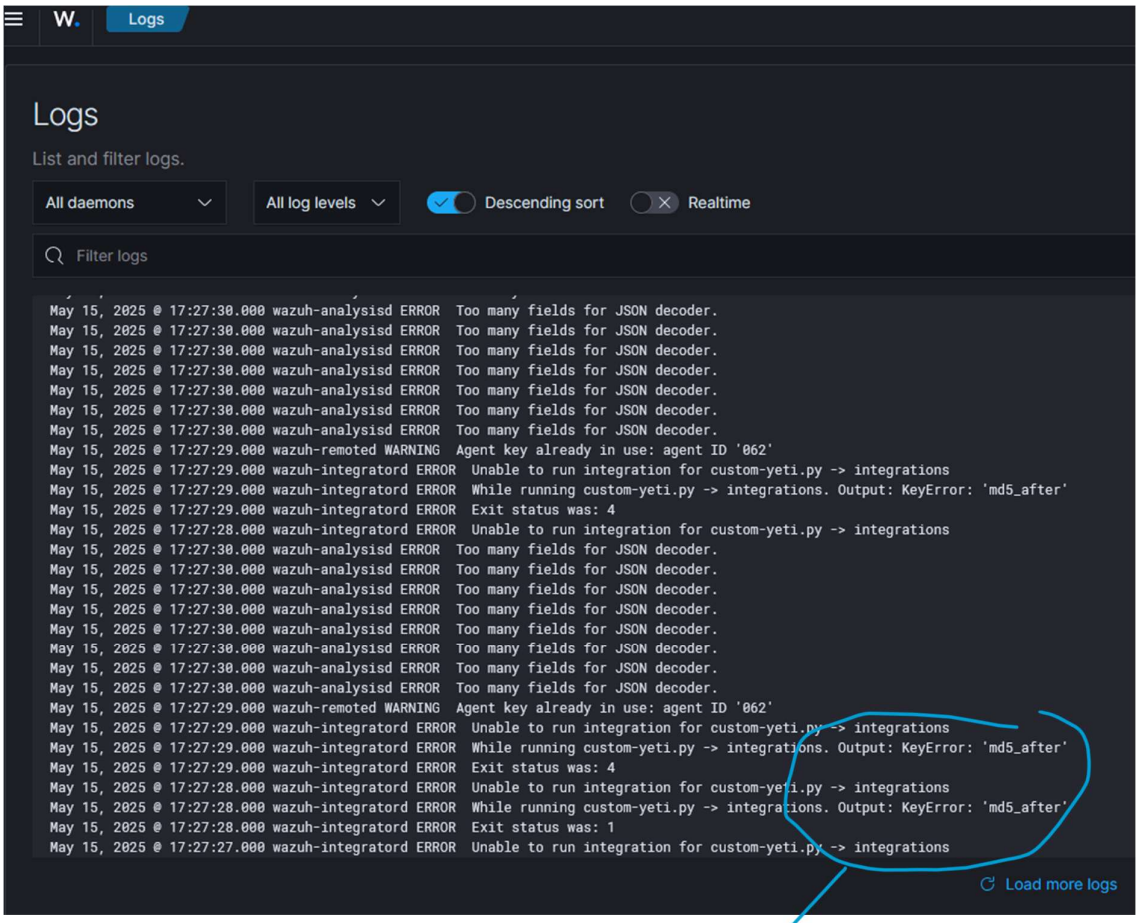
```
<directories realtime="yes">/home/<USER_NAME></directories>
```

   **IP reputation use case**

   nano /var/ossec/etc/ossec.conf di dalam <ossec_config>:
   **(cek di Lokasi log masing-masing VM dan web-server)**

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/syslog.log</location>
</localfile>
```

5. Cek hasil integrasi pada log wazuh server




```
    # Request info using Yeti API
    yeti_response_data = request_info_from_api(alert_output,
access_token, md5_hash)

    if not yeti_response_data:
        debug("No data returned from the Yeti API.")
```

```
        return None

    alert_output['yeti']['source'] = {
        'alert_id': alert['id'],
        'file': alert['syscheck']['path'],
        'md5': alert['syscheck']['md5_after'],
        'sha1': alert['syscheck']['sha1_after'],
    }
```

Ok jadi belum ada md5_hash yang teridentifikasi

**Daftar Pustaka**

1. Integrating Wazuh with Yeti platform, https://wazuh.com/blog/integrating-wazuh-with-yeti-platform/
2. Updated :Wazuh–Yeti Integration, https://medium.com/@ddigvijay/updated-wazuh-yeti-integration-5834fab6bc70
3. Git Yeti, https://github.com/yeti-platform/yeti
4. Hasil instalasi dan monitoring mandiri pada instalasi wazuh untuk Dinas Komunikasi, Informatika, Statistik dan Persandian Pemerintah Daerah Kabupaten Blitar