Реагирование на BIFIT Mitigator

?????? ?????? ???????? ????????????? KUMA Community ? ?????????????? AS IS ??? ?????-???? ???????? ? ??????????????.

Описание

?????? ????????? ??????????? ????????? ????????? ??????? (`tbl`) ? ????????, ??????? ????????? ??? ????????? ????????? (`src_ip`, `dst_ip`, `src_port`, `dst_port`, `protocol`) ??? ?? ? ????????? ????????.

Аргументы

`--server` ????????? ip ????? ???????, ????? ??????????? ? ??????? ? hostnames = ['x.x.x.x', 'y.y.y.y'] # ???????? ?????? ??????

`--user` ????????? ??? ???????????, ????? ??????????? ? ??????? ? user = 'login' # ???????? ?????

`--password` ????????? ??? ???????????, ????? ??????????? ? ??????? ? passwd = 'passwd' # ???????? ??????

`--policy` ????????? policy_id, ???? ???????? ???????

`-s`, `--ip_src` IP ????? ????????? ??? ??????????

`-d`, `--ip_dst` IP ????? ?????????? ??? ??????????

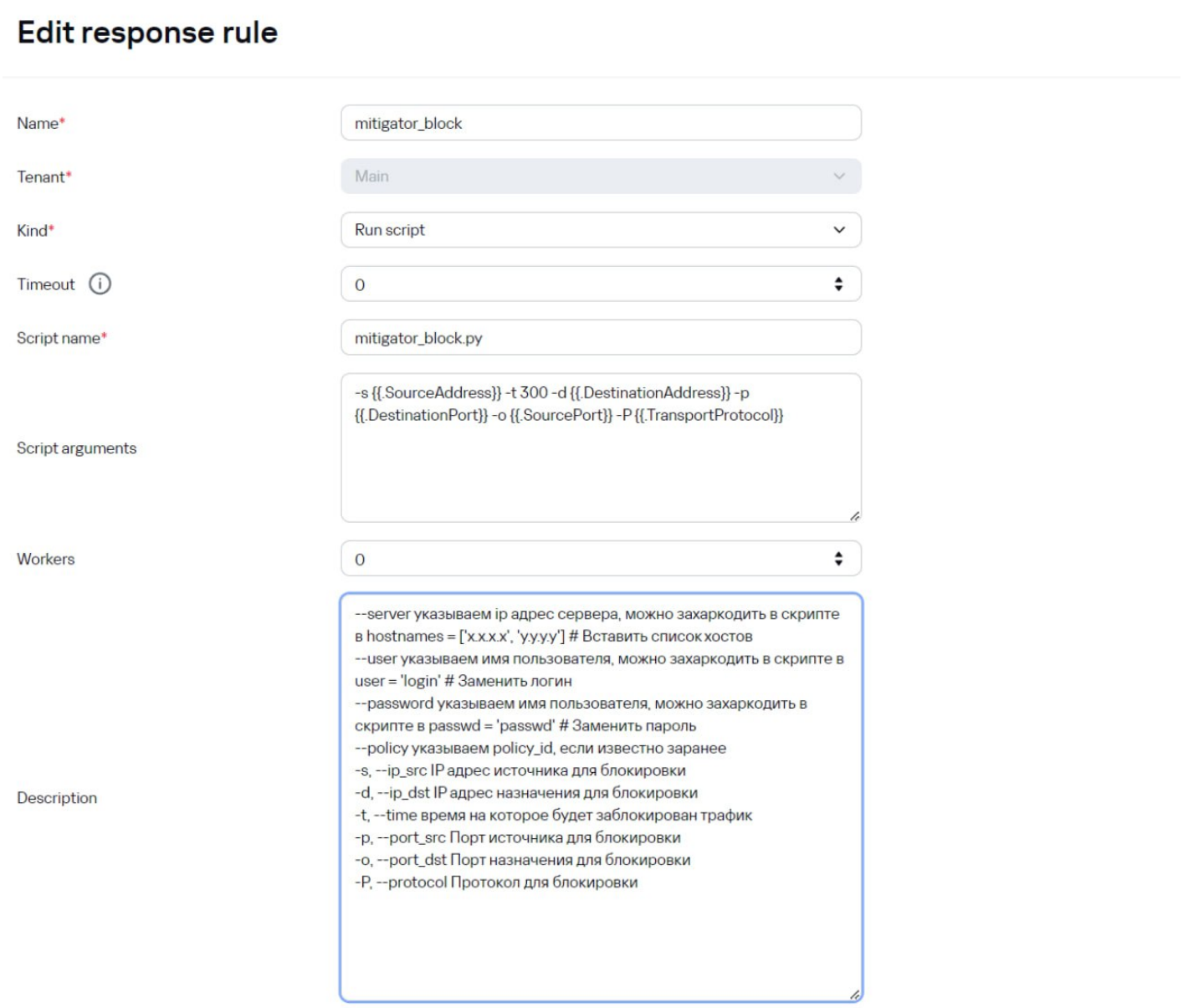
`-t`, `--time` ????? ?? ??????? ????? ??????????? ??????

`-p`, `--port_src` ???? ????????? ??? ??????????

`-o`, `--port_dst` ???? ?????????? ??? ??????????

`-P`, `--protocol` ???????? ??? ??????????

Правило реагирования



Скрипт

?????? ?? ????? python ???????? ??? ?????????

mitigator_block.py

```
#!/bin/python3

import argparse
import json
import ssl
import sys


if sys.version_info >= (3, ):
        from urllib.request import Request, urlopen
        from urllib.error import HTTPError
```

```
else:
        from urllib2 import Request, urlopen, HTTPError

class RequestEx(Request):
    def __init__(self, *args, **kwargs):
        self._method = kwargs.pop('method', None)
        Request.__init__(self, *args, **kwargs)

    # 'add_data(str)' removed in Python 3.4 in favour of 'Request.data: bytes'
    def add_data(self, data):
        if hasattr(self, 'data'):
            self.data = data.encode('utf-8')
        else:
            Request.add_data(self, data)

    # Request.method was added in Python 3.3
    def get_method(self):
        return self._method if self._method else Request.get_method(self)

class MitigatorException(BaseException):
    def __init__(self, message):
        self.message = message

    def __str__(self):
        return self.message

def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('--server', help='Mitigator host')
    parser.add_argument('--user', help='Mitigator login')
    parser.add_argument('--password', help='Mitigator password')
    parser.add_argument('--policy', help='policy ID (as shown in URL)')
    parser.add_argument('--no-verify', help='disable TLS certificate validation', action='store_true')
    parser.add_argument('-s', '--ip_src',  required=True, help='IP address to block')
    parser.add_argument('-d', '--ip_dst', required=True, help='IP address to block')
    parser.add_argument('-t', '--time', required=True, help='block time in seconds', type=int)
    parser.add_argument('-p', '--port_src',  help='Port src to block', type=int)
    parser.add_argument('-o', '--port_dst',  help='Port dst to block', type=int)
    parser.add_argument('-P', '--protocol',  help='Protocol to block')
    return parser.parse_args()

def make_request(hostname, uri, method=None, token=None, policy=None, data=None, parameters=None):

    url = f'https://{hostname}/api/v4/{uri}'

    if parameters is not None:
        url = f'https://{hostname}/api/v4/{uri}?{parameters}'

    request = RequestEx(url, method=method)
    if token is not None:
        request.add_header('X-Auth-Token', token)
    if data is not None:
        request.add_data(json.dumps(data))

    ctx = ssl.create_default_context()
    ctx.check_hostname = False
    ctx.verify_mode = ssl.CERT_NONE
    ctx = ctx
    try:
        response = urlopen(request, context=ctx)
    except HTTPError as e:
        try:
            raise MitigatorException(e.fp.read())
        except IOError:
            raise e
    return json.load(response)['data']
```

```
def block_traffic(hostname, token, options, policy_id):
#?????? ?? ????????? ?????????? ip ?????
    if policy_id:
        tbl_install = make_request(hostname,
                                f'tbl/items?policy={policy_id}&no_logs=false&source=1',
                                token=token,
                                method='POST',
                                data={
                                        "items": [
                                            {
                                                "prefix": options.ip_src,
                                                "ttl": options.time
                                            }
                                        ]
                                    } )
        tbl_install_check = make_request(hostname, f'tbl/list?policy={policy_id}', token=token,
method='GET')


def search_policy(hostname, token, options):
    data_request = ''
    if options.ip_src is not None:
        data_request += f'src_address={options.ip_src}'
    if options.ip_dst is not None:
        data_request += f'&dst_address={options.ip_dst}'
    if options.port_src is not None:
        data_request += f'&src_port={options.port_src}'
    else:
        data_request += '&src_port=88' # ???????????? ???????? ??? ??????? policy_id, ?? ? ???????? ?????
???? ?? ????????, ?????????? ????????????
    if options.port_dst is not None:
        data_request += f'&dst_port={options.port_dst}'
    else:
        data_request += '&dst_port=88' # ???????????? ???????? ??? ??????? policy_id, ?? ? ???????? ?????
???? ?? ????????, ?????????? ????????????
    if options.protocol is not None:
        data_request += f'&protocol={options.protocol}'
    else:
        data_request += '&protocol=TCP' # ???????????? ???????? ??? ??????? policy_id, ?? ? ???????? ?????
???? ?? ????????, ?????????? ????????????
    policy_request = make_request(hostname, f'policies/by_inbound_packet', token=token, method='GET',
parameters=data_request)
    policy_id = policy_request.get('policy_id')
    return policy_id

if __name__ == '__main__':
    options = parse_args()
    if options.server is not None:
        hostnames = [options.server]
    else:
        hostnames = ['x.x.x.x', 'y.y.y.y'] # ???????? ?????? ??????
    if options.user is not None:
        user = options.user
    else:
        user = 'user' # ???????? ?????
    if options.password is not None:
        passwd = options.password
    else:
        passwd = 'passwd' # ???????? ??????

    for host in hostnames:
        url_prefix = f"https://{host}/api/v4"
        try:
            data = make_request(host, 'users/session', data={'username': user, 'password': passwd})
            token = data['token']
            if options.policy is None:
```

```
                policy_id = search_policy(host, token, options)
                block_traffic(host, token, options, policy_id)
            else:
                block_traffic(host, token, options, options.policy)
        except:
            continue
```

Revision #2
Created 2024-03-04 10:07:27 UTC by Koala
Updated 2024-07-19 14:37:44 UTC by Koala