vautomator-serverless

Aug 09, 2023

VAUTOMATOR-SERVERLESS DOCUMENTATION

Note: If you are simply a consumer of the service, you DO NOT have to deploy your own instance. Assuming you have the API key, you can use the already deployed instance at https://vautomator.security.allizom.org.

If you would like to customise and/or deploy your own instance in your AWS environment, follow the below steps:

1. Install python3, node.js and aws-cli.
2. Install serverless framework: `npm install -g serverless`
3. Download the repo: `git clone https://github.com/mozilla/vautomator-serverless.git && cd vautomator-serverless`
4. Create a virtual env: `pipenv --python 3.x`
5. Install the Python requirements: `pip install -r requirements.txt`
6. Customise your `serverless.yml` file, in particular the `custom` and `provider` sections where you can specify your own S3 bucket name/SQS name/KMS key (if using Tenable.io integration, see step 6) etc. or specify multiple environments, tag your resources etc.
7. Setup your AWS profile and credentials. An account or role with at least the permissions listed in serverless.yml is required in order to deploy and run this. If you access AWS via SSO, it is recommended to install maws, and use `maws` to sign in to assume a role. `maws` will obtain the session credentials for you (depending on how you used it), save these credentials in your local AWS credential file `~/.aws/credentials`.
8. Once your AWS profile is set up, modify the `Makefile` to specify your `AWS region` and `AWS profile`. Serverless framework supports role assumption, and so does the `Makefile`, as long as your AWS config and credentials files are setup as per here, or using `maws` in the previous step.
9. *[OPTIONAL]* If you want Tenable.io support via the `/ondemand/tenablescan` endpoint (otherwise skip to step 8):
    - Create a Tenable.io user account with *standard* user permissions, and create an API key for this account.
    - Modify the top of the `Makefile` as follows:

```
AWS_PROFILE := <YOUR-AWS-PROFILE/ROLE>
# Y for Tenable.io support, N or blank if not
TENABLE_IO := Y / N

# If you would like to create a dedicated KMS for vautomator,
# specify a policy file here (an example policy file is
# provided in the repository). Otherwise leave blank if
# you would like to use default AWS SSM key for encrypted storage
KMS_POLICY_FILE := <YOUR-KMS-POLICY-JSON-FILE>

# Blank if a policy file is specified,
# or if you would like to use default AWS SSM key
KMS_KEYID := <YOUR-KMS-KEY-ID>
```

    - Once this is done, run `make setup TIOA=<Tenable-Access-Key> TIOS=<Tenable-Secret-Key>`. `TIOA` and `TIOS` are API keys generated in the first bullet point above. Based on the above values in Makefile, this will create a new or use the default AWS KMS key of your AWS account, and store the Tenable API keys in SSM in encrypted form using the KMS key.

    Note: If Tenable.io integration is desired, The most straightforward option is to specify the AWS profile and `Y` for `TENABLE_IO`, and leave other variables blank.

10. *[OPTIONAL]* Run: `make validate` to check if your `serverless.yml` is correctly formatted without deploying a stack.
11. Run `make deploy` to deploy to AWS! This will first install the required serverless plugins, then deploy the stack.
12. If you have no serverless/CloudFormation errors and if you see `Service Information` listing your lambda functions/endpoints in the output, you are good to go.

CHAPTER 1

Usage

On-demand scans are performed by invoking a handful of REST APIs. At this time, the request and response formats for most of the APIs are very simple - they expect a host as input, and return a UUID for the scan (if the host is valid). Valid host types are: FQDN, IPv4.

The REST API supports JSON.

The recommended method to use vautomator-serverless APIs in a vulnerability assessment is to use the vautomator-client.

You could use another tool such as `curl` to invoke them (see the REST API section below).

Note: At this time, all REST API endpoints are protected with an API key, which must be specified in an `X-Api-Key` `HTTP` header. If using the `vautomator-client`, this key will be retrieved by the client, provided that you are using the same AWS profile/role used to deploy `vautomator-serverless`. If not, the client will prompt you to enter an API key.

For a detailed usage of `vautomator-client`, refer to: https://github.com/mozilla/vautomator-client/blob/master/README.md

1.1 REST API

1.1.1 POST /scan

Perform all supported scans on a given host.

Parameters

- `target` is the host (FQDN or IPv4 address)

Output

- A `json` document containing step function execution name.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/scan' -d '{"target":␣
↪"www.mozilla.org"}' -H 'X-Api-Key: abcdefgh12345678'

{"executionArn":"<executionARN>:ScanAll:e9648493-9c01-11e9-85f4-874b479eba5f",
↪"startDate":1.561986763711E9}
```

Note: This is an asynchronous endpoint. Behind the scenes, the host is processed by a state machine, which invokes a number of Lambda functions to perform all scans on the host, and an email is sent to desired parties when all scans are completed and results are available.

1.1.2 POST /ondemand/portscan

Add a target to the scan queue for port scan.

Parameters

- `target` is the host (FQDN or IPv4 address)

Output

- A `json` document containing a UUID associated with the scan.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/ondemand/portscan' -d '
↪{"target": "www.mozilla.org"}' -H 'X-Api-Key: abcdefgh12345678'

{"uuid": "ac90f64c-3516-4449-bf4e-040d2f18fdc9"}
```

1.1.3 POST /ondemand/httpobservatory

Add a target to the scan queue for HTTP Observatory scan.

Parameters

- `target` is the host (FQDN or IPv4 address)

Note: While this endpoint will accept an IPv4 address, HTTP Observatory will not run a scan for an IP address only. vautomator will not complain, rather the HTTP Observatory scan results for the target will be empty.

Output

- A `json` document containing a UUID associated with the scan.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/ondemand/
httpobservatory' -d '{"target": "www.mozilla.org"}' -H 'X-Api-Key:␣
↪abcdefgh12345678'

{"uuid": "6dd38a01-4d2d-4781-8db1-3ab65b63e1fb"}
```

1.1.4 POST /ondemand/tlsobservatory

Add a target to the scan queue for TLS Observatory scan.

Parameters

- `target` is the host (FQDN or IPv4 address)

Output

- A `json` document containing a UUID associated with the scan.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/ondemand/
tlsobservatory' -d '{"target": "www.mozilla.org"}' -H 'X-Api-Key:␣
↪abcdefgh12345678'

{"uuid": "31c1f82e-83e2-4ccf-b245-8907d0a9eee8"}
```

1.1.5 POST /ondemand/sshobservatory

Add a target to the scan queue for SSH Observatory scan.

Parameters

- `target` is the host (FQDN or IPv4 address)

Output

- A `json` document containing a UUID associated with the scan.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/ondemand/
sshobservatory' -d '{"target": "www.mozilla.org"}' -H 'X-Api-Key:␣
↪abcdefgh12345678'

{"uuid": "be32e717-c72e-41d9-806f-fd4de805aae4"}
```

1.1.6 POST /ondemand/websearch

Add a target to the scan queue for a Google web search of a target with a keyword `security`.

Parameters

- `target` is the host (FQDN or IPv4 address)

Output

- A `json` document containing a UUID associated with the scan.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/ondemand/websearch' -d '
↪{"target": "www.mozilla.org"}' -H 'X-Api-Key: abcdefgh12345678'

{"uuid": "0b9e2375-1e8a-4921-8bb4-1e82f695d1dc"}
```

1.1.7 POST /ondemand/direnum

Add a target to the scan queue for a directory enumeration scan.

Parameters

- `target` is the host (FQDN or IPv4 address)

Output

- A `json` document containing a UUID associated with the scan.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/ondemand/direnum' -d '
↪{"target": "www.mozilla.org"}' -H 'X-Api-Key: abcdefgh12345678'

{"uuid": "1c124924-2938-423b-a42a-489e2dc8ac64"}
```

1.1.8 POST /ondemand/tenablescan

Add a target to the scan queue for a Tenable.io scan.

Note: This endpoint will accept submissions, however a Tenable scan will not run unless vautomator was deployed with Tenable.io support during setup (see step 7).

Parameters

- `target` is the host (FQDN or IPv4 address)

Output

- A `json` document containing a UUID associated with the scan.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/ondemand/tenablescan' -
↪d '{"target": "www.mozilla.org"}' -H 'X-Api-Key: abcdefgh12345678'

{"uuid": "a778ada0-051f-464f-bf18-599d051f0fac"}
```

1.1.9 POST /results

Downloads the scan results available for the requested host.

Parameters

- `target` is the host (FQDN or IPv4 address)

Note: In order for this endpoint to work properly, the request made must contain a `'Accept: application/gzip'` header (This is an AWS API gateway caveat).

Output

- A binary blob (`application/gzip`) containing compressed scan results for the host.

Example

```
curl -X POST 'https://vautomator.security.allizom.org/results' -d '
↪{"target": "www.mozilla.org"}' -H 'X-Api-Key: abcdefgh12345678' -H 'Accept:␣
↪application/gzip' > www.mozilla.org__results.tgz
```

CHAPTER 2

Debugging / Troubleshooting

Currently, the best way to debug or troubleshoot is to access Cloudwatch logs for your AWS account used to deploy `vautomator-serverless`.

CHAPTER 3

vautomator-serverless : deprecated

vautomator-serverless is no longer supported. The Python package that the tool used to interact with the Tenable.io API, tenable-io, was deprecated in 2020. In order to fork this project and get it working, you'd need to update vautomator-serverless to use a different Python package for interacting with Tenable.io

3.1 Original README

This project used serverless framework and attempted to create a serverless environment that could be used to automate vulnerability assessment tasks from multiple ingestion points, such as on-demand submission of a host via a REST API, regular scanning of a known list of hosts, and opportunistically scanning of hosts appearing in Certificate Transparency logs.

This is under development with more features being added as different branches. The tool currently supports:

- A single API endpoint (`/scan`) which performs all scans on a given host, and emails the results to desired email address(es).
- Addition of a target to the scan queue for port scan by an API endpoint (`/ondemand/portscan`).
- Addition of a target to the scan queue for HTTP Observatory scan by an API endpoint (`/ondemand/httpobservatory`)
- Addition of a target to the scan queue for TLS Observatory scan by an API endpoint (`/ondemand/tlsobservatory`)
- Addition of a target to the scan queue for SSH Observatory scan by an API endpoint (`/ondemand/sshobservatory`)
- Addition of a target to the scan queue for a directory enumeration scan (currently with `dirb`) by an API endpoint (`/ondemand/direnum`)
- Addition of a target to the scan queue for a Google web search by an API endpoint (`/ondemand/websearch`)
- [OPTIONAL] Addition of a target to the scan queue for a Tenable.io scan by an API endpoint (`/ondemand/tenablescan`)

- Performing requested scan type (port, HTTP Observatory, TLS Observatory or SSH Observatory) on hosts in the queue
- Scheduled port scans from a hard-coded list of hosts (disabled by default)
- Scheduled directory enumeration scans (via `dirb`) from a hard-coded list of hosts (disabled by default)
- Scheduled HTTP Observatory scans from a hard-coded list of hosts (disabled by default)
- Scheduled TLS Observatory scans from a hard-coded list of hosts (disabled by default)
- Scheduled SSH Observatory scans from a hard-coded list of hosts (disabled by default)
- An endpoint to retrieve the scan results for a given host (`/results`)
- Manually add a host to the scan queue (for PoC purposes).

All API endpoints are currently protected by an API key. Ideally this should be replaced with SSO integration.

Results from all scans are placed in an S3 bucket specified in `serverless.yml`.

Port scans are performed using a statically compiled nmap binary, packaged within the serverless application.

Directory enumeration scans are performed via `dirb`, compiled specifically for Amazon Linux and the binary and all supporting files packaged within the serverless application, similar to the `nmap` binary.

Note: UDP port scans are not supported as Lamdba functions can not run as root/privileged users.

3.2 Setup

Please refer to the setup steps here.

3.3 On-demand Scan REST APIs

Please refer to REST API documentation here.