PCAP generator

Generation of pcap files using python and docker.

Installation

Docker, docker compose and docker swarm is needed to run scenarios.

1. Install Docker on Linux: CentOS, Debian, Fedora, or Ubuntu.
2. Post installation steps do run Docker as non sudo user: https://docs.docker.com/install/linux/linux-postinstall/
3. Install Docker Compose.
4. *Optional* for Docker Swarm on multiple computers - init cluster.

Project structure

There are several folder in the project, below is a description of each.

`output` directory

Here are stored outputs from container like tcpdump, http crawler logs, selenium tests ouptput, etc. Use the directory for any files needed to be generated from containers. Include .gitignore file to write directory to git repository (but not content).

`runs` directory

Contains settings for sambla clients and server. This directory will be deleted in next release, do not use this directory.

`scenarios` directory

Here is the complete list of all implemented scenarios. In each subdirectory is usually included documentation (README.md) file, bash script for running scenario and docker-compose.yml file.

`sides` directory

This directory contains test files for Selenium IDE. Also this directory will be moved to src directory in the future.

`src` directory

To this directory belongs all source codes for containers. All containers have its own subdirectory.

`support` directory

Directory for another files which are not relevant for running containers. Put here web pages, images, or any other files.

Other files in the root directory

`generate_pdf.sh` is a bash script which generates `manual.pdf` file from this file and README.md in `scenarios` subdirectories. It uses Pandoc which is a universal document converter. `pcap_template.latex` is latex template for the output pdf.

Usage

All scenarios are in the `scenarios` directory. Scenarios can be run by `cd` to the directory and run bash script `bash run.sh`. The script runs the docker-compose file, waits 10 minutes and then ps containers and kills them.

Creating custom scenario

User can also defined own scenario by creating a configuration file docker-compose.yml. There are several parts of the configuration file - server definition, client definition and attacker definition, options for these definitions are shown in the next sections.

Here is the complete example of definition file for one scenario:

```
version: '3'

services:
  web-nginx:
    image: udfb/nginx-tcpdump
    deploy:
```

```
      resources:
        limits:
          cpus: '0.1'

    volumes:
      - ../../src/nginx/example_site:/usr/share/nginx/html
      - ../../output/tcpdump:/tcpdump
    networks:
      web:
        aliases:
          - web-nginx.example.com
    ports:
      - "80:80"
    command: sh /script.sh

  http_crawler:
    image: udfb/http-generator
    depends_on:
            - web-nginx
    volumes:
      - ../../output/http_client/log:/tmp
    networks:
      - web
    command: ["python", "run.py", "-p", "http://web-nginx.example.com"]
    deploy:
      mode: replicated
      replicas: 5

  slowloris:
    image: udfb/slowloris
    depends_on:
            - web-nginx
    networks:
      - web
    command: ["slowloris", "web-nginx.example.com"]
    deploy:
      mode: replicated
      replicas: 20

networks:
  web:
```

HTTP servers definition

Below are definitions of commonly used web servers. It is recommended to run only one server at a time.

Apache server

```
  web-apache:
    image: udfb/apache-tcpdump
    volumes:
      - ../../output/tcpdump:/tcpdump
    networks:
      web:
        aliases:
          - web-apache.example.com
    ports:
      - "80:80"
    command: sh /script.sh
```

lighttpd server

```
  web-lighttpd:
    image: udfb/lighttpd-tcpdump

    volumes:
      - ../../src/nginx/example_site:/var/www/localhost/htdocs
      - ../../output/tcpdump:/tcpdump
    networks:
      web:
        aliases:
```

```
        - web-lighttpd.example.com
    ports:
      - "80:80"
    command: sh /script.sh
```

nginx server

```
web-nginx:
    image: udfb/nginx-tcpdump
    deploy:
      resources:
        limits:
          cpus: '0.1'

    volumes:
      - ../../src/nginx/example_site:/usr/share/nginx/html
      - ../../output/tcpdump:/tcpdump
    networks:
      web:
        aliases:
          - web-nginx.example.com
    ports:
      - "80:80"
    command: sh /script.sh
```

HTTP clients definition

HTTP clients emulate normal user behavior during visiting web pages.

HTTP crawler

The script works as follows:

1. Page form the defined address is downloaded.
2. URL links in the given domain are extracted, shuffled and add to a list to visit.
3. Script waits random interval between 0 and 10 seconds, selects next address to visit and go to point 1.

There are available three backend for crawling pages:

- Headless **chrome**
- Crawler based on **urllib** library
- Crawler based on **wget** utility

All three backend tries to download not only html code of a web page but also a page content like images, css files, and javascript files.

More detailed info can be gathered by inspecting the source code.

```
  http_crawler:
    image: udfb/http-generator
    depends_on:
          - web-nginx
    volumes:
      - ../../output/http_client/log:/tmp
    networks:
      - web
    command: ["python", "run.py", "-p", "http://web-nginx.example.com"]
    deploy:
      mode: replicated
      replicas: 5
```

Selenium runner

It can run tests generated by Selenium IDE. There are available plugins for Chrome and Firefox for test recording. The recorded test can be saved saved in the `sides` directory located in the root of the project.

The tests are run once per the container startup, but scenario will run the container again until time is over.

```
  selenium-side-runner:
    image: udfb/http-selenium
    volumes:
      - ../../sides:/sides
      - ../../output/selenium:/root/out
    deploy:
```

```
    mode: replicated
    replicas: 1
```

HTTP attackers definition

Here is a list of implemented HTTP attacks using multiple vectors to corrupt normal server operations.

DoS attack

Denial of Service attack tries to use some vulnerability of protocol or its implementation to force server stop answering to requests.

```
slowloris:
  image: udfb/slowloris
  depends_on:
          - web-nginx
  networks:
    - web
  command: ["slowloris", "web-apache.example.com"]
  deploy:
    mode: replicated
    replicas: 1
```

Flood attack

Flood attack runs DDoS attacks with HTTP-flood(GET/POST).

```
http-flood:
  image: udfb/attack-http-flood
  depends_on:
          - web-nginx
  networks:
    - web
  command: ["python", "wreckuests.py", "-v", "http://web-nginx.example.com", "-t", "2"]
  deploy:
    mode: replicated
    replicas: 20
```

Scanners

Scanners try to access all resources on a server, even hidden, so they also try access non existent files.

```
http_grabber:
  image: apolloclark/kali-linux-web-cli
  depends_on:
    - web-nginx
  networks:
    - web
  command: ["fimap", "--spider", "1", "--sql", "--xss", "--url", "http://web-nginx/"]
  # https://tools.kali.org/web-applications/grabber
  deploy:
    mode: replicated
    replicas: 1
```

```
ihulk:
  image: udfb/ihulk
  depends_on:
    - web-nginx
  networks:
    - web
  command: ["python", "ihulk.py", "http://web-nginx.example.com"]
  deploy:
    mode: replicated
    replicas: 5
```

```
http_nikto:
  image: apolloclark/kali-linux-web-cli
  depends_on:
    - web-nginx
  networks:
```

```
    - web
  command: ["nikto", "-Display", "1234EP", "-o", "report.html", "-Format", "htm",
            "-Tuning", "123bde", "-ask", "no", "-Tuning", "1234567890d", "-host",
            "http://web-nginx/"]
  # https://tools.kali.org/information-gathering/nikto
  deploy:
    mode: replicated
    replicas: 1
```

```
http_skipfish:
  image: apolloclark/kali-linux-web-cli
  depends_on:
    - web-nginx
  networks:
    - web
  command: ["skipfish", "-o", "output", "http://web-nginx/"]
  # https://tools.kali.org/information-gathering/skipfish
  deploy:
    mode: replicated
    replicas: 1
```

```
http_uniscan:
  image: apolloclark/kali-linux-web-cli
  depends_on:
    - web-nginx
  networks:
    - web
  command: ["uniscan", "-u", "http://web-nginx/", "-qdwsg"]
  deploy:
    mode: replicated
    replicas: 1
```

```
http_fuzz:
  image: apolloclark/kali-linux-web-cli
  depends_on:
    - web-nginx
  networks:
    - web
  command: ["wfuzz", "-c", "-z", "file,/usr/share/wfuzz/wordlist/general/common.txt",
            "--hc", "404", "http://web-nginx/FUZZ"]
  deploy:
    mode: replicated
    replicas: 1
```

Exploiting

Exploits take advantage of some system vulnerability. This vulnerability is usually caused by developer and system administrators. Example of exploits are SQL injections, Cross Site Scripting (XSS), remote code execution, etc.

```
http_davtest:
  image: apolloclark/kali-linux-web-cli
  depends_on:
    - web-nginx
  networks:
    - web
  command: ["davtest", "-url", "http://web-nginx/"]
  # https://tools.kali.org/web-applications/davtest
  deploy:
    mode: replicated
    replicas: 1
```

```
  image: apolloclark/kali-linux-web-cli
  depends_on:
    - web-nginx
  networks:
    - web
  command: ["fimap", "-u", "http://web-nginx/"]
```

```
  # https://tools.kali.org/web-applications/fimap
  deploy:
    mode: replicated
    replicas: 1
```

```
sql_map:
  hostname: sql_map
  image: udfb/sql_map
  depends_on:
    - web-nginx
  networks:
    - web
  command: ["python", "sqlmap.py", "-u",
            "http://web-nginx.example.com/login.html?user=uzivatel&password=heslo",
            "--batch"]
  deploy:
    mode: replicated
    replicas: 1
```

SMB server definition

This SMB server is based on the Samba server which if free implementation of SMB protocol.

Server's shared folder is located in `runs/smb_server/` directory. Any other files can be added to these directories.

```
smb_server:
  # https://github.com/dperson/samba
  container_name: smb_server
  hostname: smb_server
  image: dperson/samba
  volumes:
    - ../../runs/smb_server:/shares
  environment:
  # "<name;/path>[;browse;readonly;guest;users;admins;writelist;comment]"
   - SHARE=shares;/shares;yes;no;yes;all;all,all;all share
   - WORKGROUP=workgroup
   - NMBD=true
   - RECYCLE=false
   - SMB=false
   - USER=cortex01;GreyCortex2602
```

SMB client definition

Clients source is written in Python and for access to Samba server it uses the `pysmb` package.

A client connects to server and perform one of the actions:

- list shares - every operation started with listing the available shares on the server.
- download - selects randomly one file inside randomly chosen directory and download the whole content,.
- upload - upload file in random directory which contend is randomly generated. Also name is generated randomly, but always starts with *gen_* prefix.
- delete - clients delete randomly selected file which starts with *gen_* prefix.

After performed operation, there is random wait interval between 0 and 60 seconds (except listing directory).

```
smb_client01:
  container_name: smb_client01
  hostname: smb_client01
  build:
    context: ../../src/smb/client
  environment:
   - SMB_USER=cortex01
   - PASSWORD=GreyCortex2602
   - SMB_HOSTNAME=smb_server
   - IP=smb_server
  command: ["python", "crawler_client.py", "-t", "usual", "-n", "60", "-s", "shares"]
  deploy:
    mode: replicated
    replicas: 1
  depends_on:
    - smb_server
```

SMB attacker definition

The SMB Attack client has the same operations as the SMB Client, but with different settings.

There are three types of operation settings.

- usual - see the SMB Normal chapter for more information.

  `python crawler_client.py -t usual -n 60 -s shares`

- attacker - behaves as usual clients but also tries to download non existent files defined in `src/smb/client/secret_files.txt`.

  `python crawler_client.py -t attacker -n 60 -s shares`

- attacker2 - tries to connect to server with usernames and passwords defined in `src/smb/client/users.txt` and `src/smb/client/passwords.txt`.

`python crawler_client.py -t attacker2 -n 60 -s shares`

```
  smb_attacker1:
    container_name: smb_client04
    hostname: smb_client04
    build:
      context: ../../src/smb/client
    environment:
    - SMB_USER=cortex01
    - PASSWORD=GreyCortex2602
    - SMB_HOSTNAME=smb_server
    - IP=smb_server
    command: ["python", "crawler_client.py", "-t", "attacker", "-n", "60", "-s", "shares"]
    deploy:
      mode: replicated
      replicas: 1
    depends_on:
      - smb_server
```

```
  smb_attacker2:
    container_name: smb_client04a
    hostname: smb_client04a
    build:
      context: ../../src/smb/client
    environment:
    - SMB_USER=cortex01
    - PASSWORD=GreyCortex2602
    - SMB_HOSTNAME=smb_server
    - IP=smb_server
    command: ["python", "crawler_client.py", "-t", "attacker2", "-n", "60", "-s", "shares"]
    deploy:
      mode: replicated
      replicas: 1
    depends_on:
      - smb_server
```

Run custom scenario

Manual way to run a scenario:

`docker stack deploy --compose-file docker-compose.yml webservice`

Monitor running containers:

`docker stack ps  --no-trunc webservice`

Show logs for given service:

`docker service logs webservice_[container_name]`

Terminate running scenario:

`docker stack rm webservice`

Note: in all scenarios is `webservice` name of service which containers run under.

Another docker useful commands

Create new tag (default is latest):

`docker tag mylocalimage:latest username/reponame:tag`

Example:

`docker tag udfb:nginx-tcpdump udfb/nginx-tcpdump`

Add new version to the Docker Hub:

`docker push username/reponame:tag`

Example:

`docker push udfb/nginx-tcpdump`

Shows log for given service:

`docker service logs webservice_http_fuzz`

Generate pdf version of this file:

https://www.markdowntopdf.com/

or directly from github,

or just install Pandoc and run generate_pdf.sh to generate whole documentation:

```
bash generate_pdf.sh
```