Automated Network Packet Generation for Evaluation of Neural Networks in Intrusion Prevention Systems

Bernhard Gally
IT Security
FH Technikum Wien
Vienna, Austria
cs19m023@technikum-wien.at

Sebastian Lipp
IT Security
FH Technikum Wien
Vienna, Austria
cs19m032@technikum-wien.at

Damir Marijanovic
IT Security
FH Technikum Wien
Vienna, Austria
cs19m031@technikum-wien.at

***Abstract*—@Damir**

test

***Index Terms*—artificial intelligence, neural networks, computer networks, network security, packet generation**

I. Introduction

@Damir [1p]

II. Infrastructure

This chapter covers the design of the infrastructure to provide a tool for creating generic environments in which specific network scenarios can be simulated to record network packets and extract features with which neural networks inside intrusion prevention systems could be trained and evaluated.

Therefore it is split into four parts. The first part designs the fundamental simulation architecture. The second part focuses on the creation of the hosts, followed by the configuration of the network in the third part. The last part covers the automation of the procedure with the help of provioning tools.

A. Designing the architecture

A simulation network consists of multiple scenarios based on network attacks and benign traffic. In this network generic hosts can be placed and connected to each other or the internet with the usage of hubs, switches and routers. This hosts reflect attackers and victims, or usual clients and servers, as well as recorders. To ensure recorded features can be labelled, recorders are placed on all positions where another kind of attack or network traffic should be recorded. Furthermore they are able to filter out selected traffic to ensure only the important traffic is recorded. This means network traffic is recorded within multiple files marked as bad or good which serve as input for the data extractor to extract the features and create the dataset to train and test the neural network.

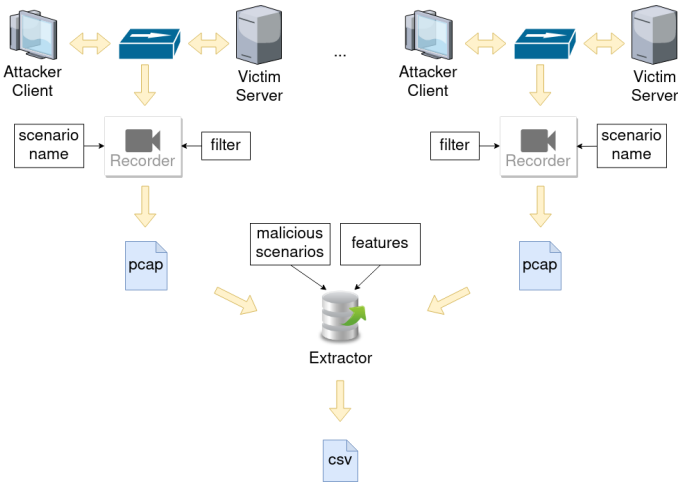

Fig. 1. Design principle

Fig. 1 shows the design principle. The recorders receive the data through network hubs placed between the attackers and victims, or clients and servers and record in pcap files to a central filesystem. The scenario names and filter settings are passed to the recorders as parameters. The recorded files - named like the associated scenario - are then collected and forwarded to the extractor which knows the names of the malicious scenarios and extracts the selected features to a csv file. The extractor is only required once and is not part of the simulation environment.

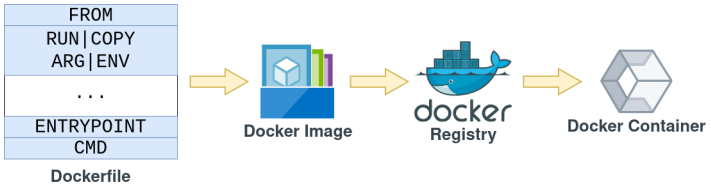

Fig. 2. Node creation

B. Creating the nodes

To offer users a very generic approach Docker containers are used for creating customized network nodes. The configuration of the nodes is based on code and can be easily created. It is defined within Dockerfiles which can be transformed to an image and uploaded to a registry through Docker client tools as shown in Fig. 2. When the images are placed on a public Docker registry, they can be pulled and executed from any place. This is the default approach in the current infrastructure design.

The lines in a Dockerfile are describing layers of which the image consists. The first line is a statement to to specify the base image, for example Alpine Linux or Ubuntu. The new image inherites everything from this base layer. After that multiple commands can be added to install software, configure the system, copy files to the image or define environment variables. The last lines define the entrypoint and/or a command which the container runs when it is started. This could be for instance, a program to attack a host, or record the network traffic. Docker also supports mounting volumes to the containers when they are started as well as passing arguments to use the same image for similar nodes with different settings. More about creating Dockerfiles can be found on the offical documentation of Docker.[1]

Once the images are built and uploaded to an accessible registry they can be further used in the simulation environment. The target hosts only need the Docker runtime environment installed and access to the image registry. In this case the official Docker Hub is used to build and store the images for the scenarios. This includes all clients and servers, the recorder and the extractor.

The network recorder is based on a customized Docker image which runs tcpdump with additional arguments like the filename to specify the scenario name and the filter expression to sort out unwanted traffic. It records the packets passing the first interface. For more information about possible commands and configuration refer to the manpage of tcpdump.[2]

The extractor more precisely the CI flowmeter is also placed into a Docker container mounting a volume containing all the recorded pcap files. It extracts only the features defined by a parameter and saves the csv file with the extracted features at the same place as the pcap files which can be later transferred to another host to evaluate the data.

[1] https://docs.docker.com/develop/develop-images/dockerfile_best-practices
[2] https://www.tcpdump.org/manpages/tcpdump.1.html

The runtime configuration and connection of the network nodes as well as the creation of other nodes like switches, hubs and routers is part of the simulation environment which is covered in the next section.

C. Connecting the nodes

The configuration of the network, more precisely the connection between the network nodes is done within GNS3, an open-source network simulator which can be easily controlled via a native graphical user interface or RESTful API. It consists of a server and a client application. In this setup the server application is installed on a virtual machine which offers 8-cores, 32 GiB RAM and 300 GiB hard disk space. It is possible to install multiple servers but in this case one was sufficient for simulating multiple scenarios including DoS attacks. More information about the installation of the server can be found on the official website of the GNS3 project.[3]

Once the server runs a project can be created and the scenarios can be configured via the graphical interface of the GNS3 GUI. It is recommended to setup a VPN to the server to ensure the terminals of the network nodes are reachable remotely without configuring network tunnels. This means it is necessary to connect to the server via VPN before the GUI can reach the server applicaiton.

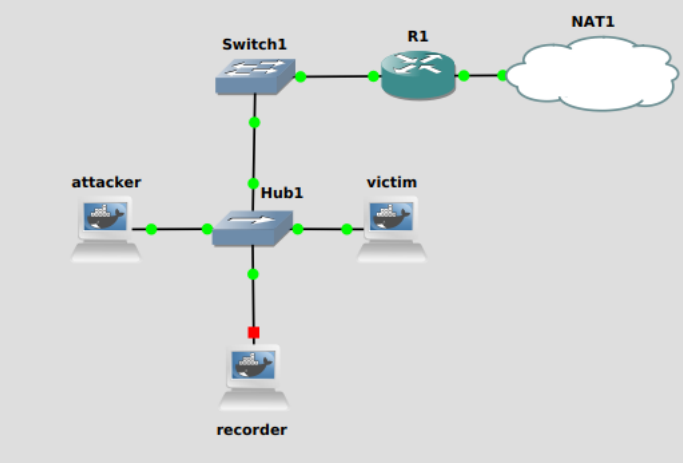

Fig. 3. Network example

Fig. 3 shows a simple network created in GNS3. The attacker, victim and recorder are created through Docker templates referring the created Docker images before. GNS3 supports the configuration of environment variables and volumes for Docker containers. All other components including switches, routers and the interface to the internet are basic network components of GNS3 and are already available in the device browser. Cisco routers need official IOS images before they can be created.

The project is automatically saved to the filesystem and includes all components which are needed to run the network including configuration, logs and recorded data. The simulation can be started through the GUI or the API and produces the pcap files within the project directory. Once the simulation is stopped the pcap files can be copied to a temporary directory and send through the extractor container to produce the csv file including the extracted features.

Fig. 4. Project structure

The setup of the network is done through the GUI because the API does not support all features to create network nodes. The project folder can be checked in into version control and copied to the server when it is needed. As shown in Fig. 4 it consists of the gns3 file which contains all the network configuration in a JSON format. The project-files directory contains additional files which are part of the filesystems of network nodes for example the docker containers. The text files are the parameters which are passed to the extractor to determine the attack names and the features to extract.

D. Automating the procedure

The automated setup of the server and run of the simulation is done within Anisble playbooks. Therefore multiple roles are created, one for the setup of the GNS3 server, one for running the network simulation and one for extracting the features. As shown in Fig. 5 the server role is associated with the setup playbook, while the simulation and extraction roles are used by the run playbook. These playbooks are stored in YAMl files and can be executed with the ansible tools.[4]

The server role set ups the GNS3 server and OpenVPN, copies the device images and projects and configures the Docker runtime environment. It downloads the VPN configuration to the client which can be later used to connect to the server. Because the test setup restricted the incoming ports, VPN was changed to use TCP instead of UDP to enable the functionality of creating SSH tunnels.

To run the simulation role a VPN connection to the server is required. This is accomplished by a simple connection script which opens a SSH tunnel and connects to the server via VPN.

[3]https://docs.gns3.com/1c2Iyiczy6efnv-TS_4Hc7p11gn03-ytz9ukgwFfckDk/index.html

[4]https://docs.ansible.com/ansible/latest/user_guide/playbooks.html#working-with-playbooks

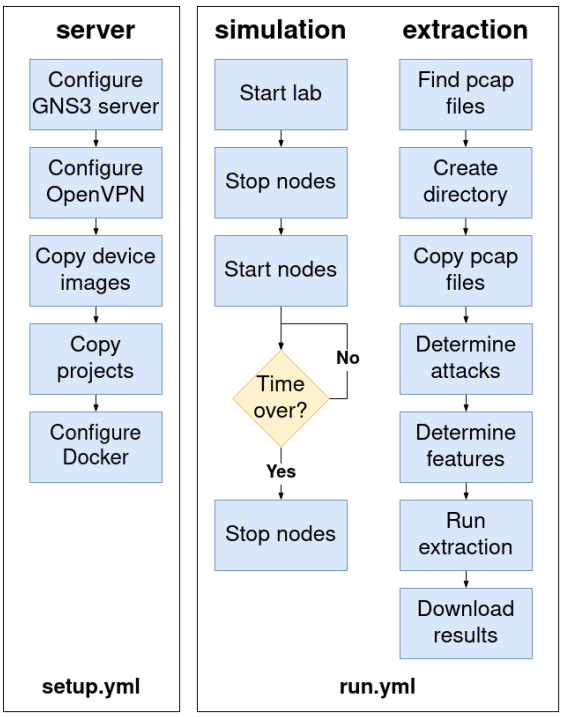

Fig. 5. Ansible roles

This is necessary because Ansible needs access to the GNS3 API and the network components. When the role is executed it opens the project, stops all network nodes and starts them again. The duration can be defined through a parameter. When the time is over the nodes are stopped. To control GNS3 via the API an additional Ansible collection is used. [5]

The extraction role gathers the pcap files, creates a temporary directory and copies all files there. Then it determines all attack scenarios and features from a file in the project directory and runs the extraction container with the mounted pcap volume and the passed parameters. After that it downloads the csv file with the extracted features. If specified in a parameter the pcap files are also downloaded.

The server can be setup by running

```
ansible-playbook -i hosts setup.yml
                 --ask-become-pass
```

Once the GNS3 projects are checked in into the Ansible project folder the simulation can be started with

```
./connect.sh
ansible-playbook -i hosts run.yml
                 --ask-become-pass
                 -e project=<project_name>
                 -e duration=<minutes>
                 [-e pcap=true]
```

All Ansible commands have to be executed within the gns3 folder and need the sudo password from the server user. The hosts file contains the target hosts configuration.

[5]https://github.com/davidban77/ansible-collection-gns3

The next chapter focuses on the implementation and simulation of specific scenarios within a GNS3 project with the help of the introduced infrastructure to produce a new dataset for evaluating neural networks in IPS systems.

III. Simulation

@Bernhard [2-3p]

IV. Conclusion

@Damir [0.5p]

References

[1] zzz