09-08-2022

Deliverable D8.10
Firewall on Demand

Deliverable D8.10

Contractual Date:	31-08-2022
Actual Date:	09-08-2022
Grant Agreement No.:	856726
Work Package	WP8
Task Item:	Task 3.4 FoD
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	LRZ/DFN
Document ID:	GN4-3-22-DD6ABC
Authors:	David Schmitz (LRZ/DFN)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).

Abstract
This document provides a summary of the enhancements and improvements to Firewall on Demand (FoD) developed and tested during GN4-3. These comprise mainly development of the core service as well as the introduction of Docker containers for installation and runtime-management support and the establishment of testing on various layers.

Table of Contents

Table of Figures

Executive Summary

The aim of this report is to provide a summary of all enhancements and improvements to Firewall on Demand (FoD) developed and tested within the current phase of the GÉANT project (GN4-3).

Firewall on Demand is a Distributed Denial of Service (DDoS) mitigation solution which was developed in earlier phases of the GÉANT project [GN_FoD_Svc]. It is currently provided in the GÉANT core network as a production service to allow National Research and Education Network (NREN) Network Operations Centre (NOC) administrators to administer DDoS mitigation rules via a web interface. It enables them to block malicious traffic within the GÉANT core network towards their NREN network without having to have actual access to the core network or to contact GÉANT NOC administrators.

In addition to the production service for GÉANT core network traffic, FoD server software is available free of charge under the terms of the GNU General Public Licence to be installed and run by anyone, specifically targeted at other NREN NOCs.

Significant enhancements and improvements to FoD developed and tested during GN4-3 include:

- The application was fully ported from Python 2 to Python 3.
- The REST API was enhanced, including for cooperation and coupling with NeMo, GÉANT's DDoS and traffic anomaly detection and analysis tool.
- Support for rules with IPv6 addresses was added.
- Use of Docker containers was introduced for installation and runtime-management support.
- Automated testing of the web user interface and REST API and integration testing with a virtual attacker/victim router/network were added.

Subject to any findings and recommendations arising from the Product Lifecycle Management (PLM) process, piloting of these enhancements and improvements is expected to take place in the second half of 2022, with deployment into production before the end of the year. Meanwhile work to extend these areas further is also planned.

Firewall on Demand continues to be a key component of GÉANT's Network Security Handling and Response Process offering, with the NREN users providing valuable input towards identifying and prioritising areas for development, ensuring that it addresses real needs and remains relevant, useful and used.

1 Introduction

Firewall on Demand (FoD) [GN_FoD_Svc] is a Distributed Denial of Service (DDoS) mitigation solution based on Border Gateway Protocol (BGP) FlowSpec [RFC_5575]. It was developed in earlier phases of the GÉANT project[1] and is currently provided in the GÉANT core network as a production service to allow users (National Research and Education Network (NREN) Network Operations Centre (NOC) administrators) to administer BGP FlowSpec rules via a web interface [GN_FoD_Iface]. This allows them to filter normally routed GÉANT IP traffic based on the administered BGP FlowSpec rules [RFC_5575] [RFC_8955]. FoD is a key component of GÉANT's Network Security Handling and Response Process (NSHaRP), the umbrella for the network security-related capabilities offered to GÉANT NRENs [NSHaRP]. At the time of writing, FoD is being used by 28 NRENs, plus CERN.

In addition to the production service for GÉANT core network traffic, FoD server software is available free of charge under the terms of the GNU General Public Licence to be installed and run by anyone, specifically targeted at other NREN NOCs [GN_FoD_GitHub].

This document provides a summary of all enhancements and improvements to Firewall on Demand developed and tested by Work Package 8 Security, Task 3 Products and Services: Firewall on Demand (WP8 T3.4) within the current phase of the GÉANT project (GN4-3). These comprise mainly actual development of the core service as well as the introduction of Docker containers for installation and runtime-management support and the establishment of testing on various layers. To help prioritise the work and reflect the needs of users, a user study was performed. This consisted of individual interviews with selected FoD service key users, based on a list of questions covering the main aspects of all the potential enhancements. A summary of the questions and an analysis of the results are provided in Milestone M8.4 *FoD Future Development Requirements Analysis* [M8.4].

This document is structured as follows:

- Section 2 covers all improvements and enhancements regarding development of the core service.
- Section 3 describes the use of Docker containers for installation and runtime-management support and for the establishment of testing on various layers.
- Section 4 covers a demonstration that is possible with some of the testing means that have been developed.

[1] FoD service development was started by GRNET during GN3plus (April 2013 to March 2015) and entered production as Version 1.1.1 during GN4-1 (May 2015 to April 2016). A summary of FoD development in the previous phase of the GÉANT project, GN4-2 (May 2016 to December 2018), is provided in the GN4-3 Milestone M8.4 *FoD Future Development Requirements Analysis* [M8.4], which in turn references the following GN4-2 deliverables: D8.2 *Firewall-on-Demand Progress Report* [GN4-2_D8.2], D8.3 *Distributed Denial of Service Mitigation v1.0 Pilot* [GN4-2_D8.3] and D8.12 *Distributed Denial of Service Mitigation v1.0 Pilot Follow-up* [GN4-2_D8.12].

- Section 5 summarises various further enhancements made.
- The last section, 6, draws together the key points of the document and outlines plans for the near- and long-term future.

2 Service Development

This section covers all development carried out for the improvement and enhancement of the Firewall on Demand core service. At the start of GN4-3, the version of FoD in production was v1.5; the improved, enhanced version is v1.7. The improvements and enhancements can be grouped into three areas, namely:

- Rewrite of FoD for Python 3 (Section 2.1).
- REST API improvements, in particular for use by NeMo (Section 2.2).
- All other improvements and enhancements by development (Section 2.3).

2.1 Rewrite of FoD for Python 3

One of the main areas for the development of FoD during GN4-3 was the adaptation and partial rewrite of the FoD application to switch from Python 2 to Python 3.

The code of FoD existing at the start of GN4-3 was based on Python 2 / Django 1.4 since it was originally developed in earlier project phases in about 2015. Python 2 was deprecated at the start of the current project phase and about to be unsupported in 2020, so upgrading the FoD code base to Python 3 / Django 2 was a high priority and was begun in the middle of 2019.

During the course of upgrading the code, it became more and more apparent that some sections of the existing code required full replacement and reimplementation. This was not so much due to differences on the syntax level between Python 2 and Python 3, but rather to major changes in the Python dependencies available for Python 3 in contrast to those available for Python 2, and resulting incompatibilities in Python dependencies used by the old FoD code. Some of these dependencies changed quite extensively and others even had no direct successor, mainly Celery [Celery] and django-registration [django-reg].

Both the old and the new FoD are split into two components, the web server part based on Gunicorn [Gunicorn] and the back-end part based on Celery [Celery] for actual coordinated execution of tasks such as NETCONF updates and SNMP querying. Between the two components, a task broker, running as a separate application, is used, which has to be compatible with the Python dependency Celery. In the old FoD the broker Beanstalk was used [Beanstalk]. In the Python 3 Celery dependency, this broker is no longer supported, requiring a switch to another broker. For this reason the broker Redis was chosen [Redis]. One advantage of this was that the old self-implemented message-queueing and -polling functionality used for the user interface's live status console could be reimplemented based on Redis message-queueing functionality.

The support for the Shibboleth [Shibboleth] registration based on django-registration [django-reg] also required some reimplementation to replace the functionality no longer provided by django-registration. Shibboleth is used to register and to authenticate each user, so that only registered and approved users, clearly assigned to the correct NREN network in FoD for multi-tenancy, can edit mitigation rules for their NREN network.

The main changes related to the switch to Python 3 can be summarised as follows:

- Generally, translate all Python 2 language constructs to Python 3 equivalents, where necessary.
- Replace Beanstalk with Redis as a task broker and adapt FoD's code accordingly.
- Rewrite the message queuing for the live-status functionality completely.
- Replace and add code to support Shibboleth registration under the new django-registration framework.
- Further adaptations necessary because of switching to newer versions of other required Python dependencies.

2.2 Improvements to the REST API

Another main area for the development of FoD was the enhancement of the rule control REST API for FoD, which enables the automated querying, creation, changing and deletion of rules. The rule control REST API, which was based on an implementation provided by the original developers of FoD, required some enhancements to make it easier to use and to allow a greater degree of atomicity of the REST transactions. Because of these changes, the two versions of the API are not fully compatible.[2]

This was for general use by users of FoD as an alternative to access via the web user interface (UI), as well as especially for the envisioned and planned coupling of FoD and the DDoS mitigation service NeMo [NeMo], supported and covered by WP8 T3.3.

The old implementation of the API (FoD v1.5) has therefore been enhanced (FoD v1.7) to:

- Allow greater degree of atomicity of REST API transactions, without the need to resolve static IDs, e.g., for IP protocols, with a separate REST API call before the actual REST API call of an explicit create/update rule action.
- Extend REST API transactions beyond the mimicking of possible actions available in the UI:
  - Introduce a parameters scope and username which allow an admin account to query rules from the point of view of a particular user.
  - Allow the creation of rules in INACTIVE state.
  - Remove restriction regarding the uniqueness of source/destination prefix match conditions.
  - Allow full deletion of rules vs. setting a rule status to INACTIVE.
- Allow querying of mitigation statistics by a REST API call.

[2] There is not yet a way to handle API calls conforming to the old API for the new API in v1.7. Users of the API will have to change their code. However, this will have a minimal impact on users, as to the team's knowledge there are no active users of the API in the current production version of FoD (v1.5).

2.3 Other Code Improvements and Enhancements

Beyond the improvements and enhancements of FoD code covered in the two previous sections, several other code-related improvements and enhancements were developed.

The two open issues from 2019, discovered after the production deployment of FoD v1.5, were addressed, namely:

- Statistics graphs were not created for mitigation rules which used IP fragmentation filter flags.
  Fixes implemented.
- eduGAIN login errors are cumbersome and do not really indicate the actual error. This was a pre-existing issue not specific to v1.5. The relevant error messages were hard-coded into FoD's code and did not take into account the possibility of flexibly and generically changing eduGAIN Shibboleth authentication attributes in FoD's configuration settings.
  Shibboleth login error messages fixed to reflect the actual FoD configuration and not be static to a particular default case.

Both issues had only been partially investigated and therefore not fully fixed and tested at the end of GN4-2. For this reason, the first months of GN4-3 were devoted to addressing these issues.

In addition:

- The display of eduGAIN user names in the web UI was improved to replace the username with given name and surname instead of the sometimes cryptic-looking user identifier.
- The possibility to prolong the active status / expiration period of rules via the UI and REST API, without interruption of the actual FlowSpec mitigation, was added.
- Support for mitigation rules using IPv6 address prefix match parameters was added. The support of FlowSpec rules using IPv6 addresses as match parameters requires the use of a NETCONF YANG model that differs from the YANG model used for rules using IPv4 addresses. After investigating this, parallel support of both YANG models in FoD was realised, so that FlowSpec rules deployed by FoD can now use either IPv6 addresses or IPv4 rules.
- For easier coordination and differentiation between different tools, services and other means of deploying BGP FlowSpec rules to the routers, it is now possible to configure a statically added rule name prefix in FoD transparently used for every rule deployed by FoD.

2.4 Status

All of these improvements and enhancements have undergone extensive developer and integration testing, including using the layered testing described in Section 3.2. At the time of writing (early August 2022), testing in the production network is underway, with piloting expected to take place in the second half of 2022 and deployment into production before the end of the year.

3 Introduction of Docker Containers

During the course of GN4-3, the possible use of Docker containers, for running FoD itself as well as for performing testing on different layers of FoD, was introduced and enhanced.

3.1 Installation and Runtime-Management Support

Beyond the support for the GÉANT network FoD service, installation support for FoD in other NRENs was always somewhat limited due to insufficient time to properly reproduce errors reported by other users who were testing the FoD installation within their own environments.

To overcome this, Docker-based containers for FoD installation were introduced. This provides users of the software with a fully working container-based reference installation of FoD which they can use either to compare and fix their own installation trials, or even directly as a container-based FoD instance. Docker containers for FoD reference and test installation under either CentOS or Ubuntu/Debian are available.

In general, installation support in Docker, as well as directly on the host system, was added, including integration with systemd [systemd], email notifications on start-up errors, and FoD runtime status.

Beyond running FoD itself in Docker, testing of FoD is supported by various Docker containers, which is outlined in the following section.

3.2 Establishment of Testing on Various Layers

Various layers of FoD testing were introduced and enhanced during the course of GN4-3, especially supported to run in Docker containers.

The layers of testing are:

- Automated web UI testing with Selenium (an umbrella project for a range of tools and libraries that enable and support the automation of web browsers [Selenium]).
- Automated REST API testing with pytest (a full-featured Python testing tool [pytest]).
- Integration testing with NETCONF DB server and a virtual attacker/victim router and network using Mininet (a network emulator tool [Mininet]).

Automated web UI testing

The use of Selenium [Selenium] web UI testing was introduced and continuously enhanced and extended over time. Many aspects of the web UI application are now covered by several Selenium tests, including virtually all buttons in all of the web pages of the web application. The whole test suite can be run in a fully automated manner in an extra Docker container. Selenium testing is useful not only for general checking of the web UI functionality but also – in a fully automated manner – for regression testing during development.

Automated REST API testing

As a complement to Selenium testing for the web UI, automated (regression) testing for the REST API was approached directly in Python based on the pytest framework [pytest].

Currently, it tests the main aspects of the REST API, i.e. creation, editing and deletion of a single rule for different combinations of traffic-matching parameters and mitigation actions. In the future, a more comprehensive scope covering each and every potential parameter combination should be approached; based on that, comprehensive coverage of REST code in the server can be reached. Also, stress tests involving the creation of a greater number of active rules might be considered.

Integration testing with NETCONF DB server and a virtual attacker/victim router and network

Furthermore, the possibility of testing FoD without an actual BGP FlowSpec-capable router was introduced. At first level, there is now a Docker container which runs a NETCONF DB server with appropriate FlowSpec YANG models to be used with FoD as a simulated router (shown in Figure 4.1 below). This way, FoD tests can be carried out to create/edit/delete FlowSpec rules to be stored within the NETCONF DB server only, without any external effects. Two implementations of NETCONF DB servers are supported: Netopeer2 [Netopeer2], developed by CESNET, and netconfd [netconfd], available as a Debian/Ubuntu package.

At second level, based on the NETCONF DB server container, an extended container (vnet2 container) providing a virtual attacker/victim router and network based on Mininet [Mininet] for use with FoD was compiled (shown in Figure 4.2 below). In this container, Mininet provides virtual attacker and victim hosts and OpenFlow switches representing virtual routers. A polling loop maps FlowSpec rules in the NETCONF DB to OpenFlow rules on the OpenFlow switches in Mininet. In turn, the OpenFlow statistic counters of the virtual routers are mapped via a Perl script inside a running snmpd daemon to SNMP statistics in the Junos-specific mitigation statistics Management Information Base (MIB), so that FoD can again query them. Virtual attacker test traffic can be created from one virtual host to another inside Mininet and so the mitigation and their statistic counter on the virtual routers can be simulated and tested.

At third level, a further extension to the NETCONF DB container (vnet2-exabgp container) adds demonstration and testing with ExaBGP [ExaBGP] and the BIRD v2 [BIRD] routing daemon, to emulate BGP-FlowSpec-capable router hardware that does not support NETCONF (shown in Figure 4.3 below). Here, the polling loop maps FlowSpec rules in the NETCONF DB to BGP FlowSpec rules via ExaBGP to a BIRD v2 instance on a virtual host in Mininet mimicking a router. To complete the demonstration, this BIRD v2 instance is connected via BGP to other virtual hosts in Mininet also running BIRD v2 and so is exchanging its BGP FlowSpec rules with them. Mitigation is not supported in BIRD v2 itself, so this aspect is not really covered in this type of demonstration.

4 Demonstration of the FoD Mitigation Workflow

The virtual attacker/victim network container (vnet2 container) introduced in the previous section can be used to demonstrate FoD's mitigation workflow even without actual router hardware.

Such a demonstration facilitates developer testing, allowing it to be done even on the isolated, local development machine of a single developer or, more generally, when no testing hardware is available at all. The demonstration can also be used by potential users to easily test FoD installation and functionality without needing to have and configure an appropriate router.

This section covers the demonstration in more detail.

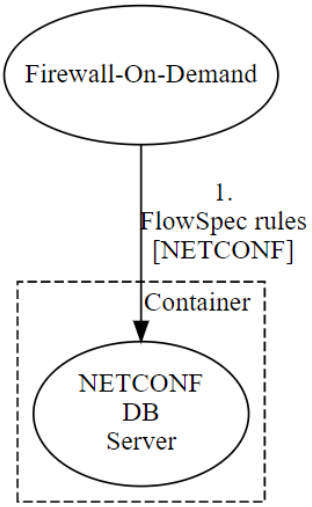

Figure 4.1: Demonstration of FoD mitigation workflow without router hardware (NETCONF database server only)

Figure 4.1 shows a possible first-level demonstration of FoD mitigation workflow without router hardware where only a NETCONF database server running in a Docker container is used. In contrast, the extended and more advanced second-level possibility using the virtual attacker/victim network (vnet2-container) is shown in Figure 4.2 and described in detail in the following.

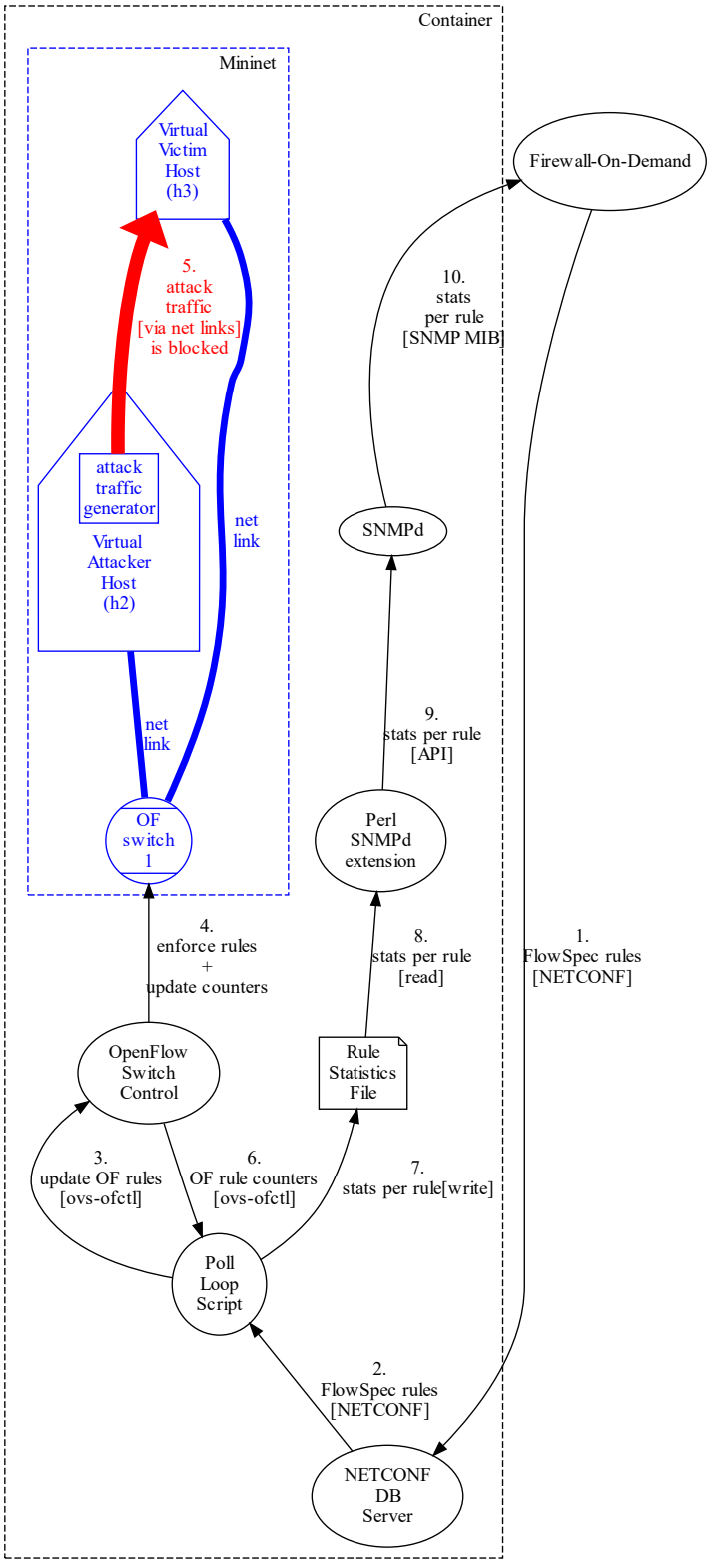

Figure 4.2: Demonstration of FoD mitigation workflow without router hardware (with virtual attacker/victim network and actual mitigation, yet no BGP involved)

DDoS attack traffic from one virtual host to another virtual host can be represented by appropriate network traffic generated on a virtual host, called “demo attack” in the following. Potential candidate tools for generating such traffic are, for example, ping [ping], iPerf [iPerf] and hping3 [hping3]. Other tools have not been tested yet, but are possible as well. In the future, more sophisticated simulation of DDoS traffic could be performed, for example, by replaying captured real DDoS traffic, or, more generally, by involving more than one attacker host in the scenario. This depends on the actual Mininet network topology used. For a simple case of a single attacker and a single victim, a Mininet topology consisting of two virtual hosts (attacker and victim) and a single OpenFlow switch in between is sufficient. Figure 4.2 shows this simple Mininet topology. Step 5 in Figure 4.2 represents the blocking of the demo attack. Options for specifying the Mininet topology (actually, any Mininet command option) can be given at the start of the vnet2 container.

Before any mitigation is applied in FoD regarding the demo attack, the attack traffic should reach the victim host. In the case of ping, iPerf or hping3, for example, this can be seen by the command output itself. Otherwise, the attack reaching the victim could be verified on the virtual victim host itself, e.g., by the tcpdump [tcpdump] or TShark [TShark] tool. Alternatively, for verification of the demo attack, tcpdump or TShark can be run on the interface of the virtual OpenFlow switch interconnecting the attacker and victim host.

When the mitigation of this demo attack is applied in FoD (in the UI or via REST) by adding an appropriate FlowSpec rule to block the traffic, FoD injects the rule via NETCONF into the NETCONF DB server (step 1 in Figure 4.2); the poll loop regularly querying the NETCONF DB server finds the new rule (step 2) and translates it to an OpenFlow *DROP* rule on any of the OpenFlow switches in the Mininet (step 3). The OpenFlow switch implementation used by Mininet actually enforces the filtering rules (step 4). The support for the mapping of FlowSpec rules in NETCONF to OpenFlow rules is only partial due to differences in the matching capabilities of FlowSpec and OpenFlow. IP prefixes, IP protocols, and TCP/UDP exact port matches are supported. Matching of, e.g., arbitrary TCP/UDP port ranges, TCP flags, and IP fragmentation flags are not supported yet. The former might be possible by mapping a single TCP/UDP port range to multiple TCP/UDP bit mask matches corresponding to the port range, but resulting in multiple OpenFlow rules. Alternatively, the emulation of mitigation could be based on IPtable rules. Similarly, only drop mitigation is supported as yet, not rate-limiting. However, for demonstration purposes the current coverage of FlowSpec rules is sufficient.

The poll loop querying for any active rule in the OpenFlow switches also queries the byte and packet counters of matched, i.e., dropped, traffic (step 6) and stores this information in a status file (step 7). In addition, snmpd [snmpd] is running in the container (outside of Mininet) and has configured an SNMP extension, written in Perl, responsible for generating a MIB subtree for the mitigation statistics in the way expected by FoD, i.e., compatible with the Juniper mitigation statistics MIB subtree. Actually, for each OpenFlow rule and byte/packet counter, the counter values of all OpenFlow switches are summed up and provided in this MIB subtree as a single value for the byte and packet counter. This MIB subtree is queried by FoD, as it would be in the case of a Juniper-compatible router, to yield the mitigation statistics of the respective rule. In contrast to a real-world scenario with real Juniper router hardware, only this single SNMP instance is queried instead of querying multiple real routers and summing up the statistics values inside FoD. For the FoD user the result is the same, as FoD currently provides only summed-up statistics data, not differentiating between the single mitigation counter values of the individual routers. In the FoD UI, the mitigation statistics collected and shown can be seen by the user – as normal – and so are an indication of a successful mitigation of the demo attack.

FlowSpec rules are ultimately exchanged as special BGP routes on the router level. This aspect of the FoD mitigation workflow is not covered by the testing containers described above. Accordingly, the further extended container, vnet2-exabgp, was compiled to provide a demonstration of BGP communication as part of the FoD mitigation workflow, as represented in Figure 4.3 below.

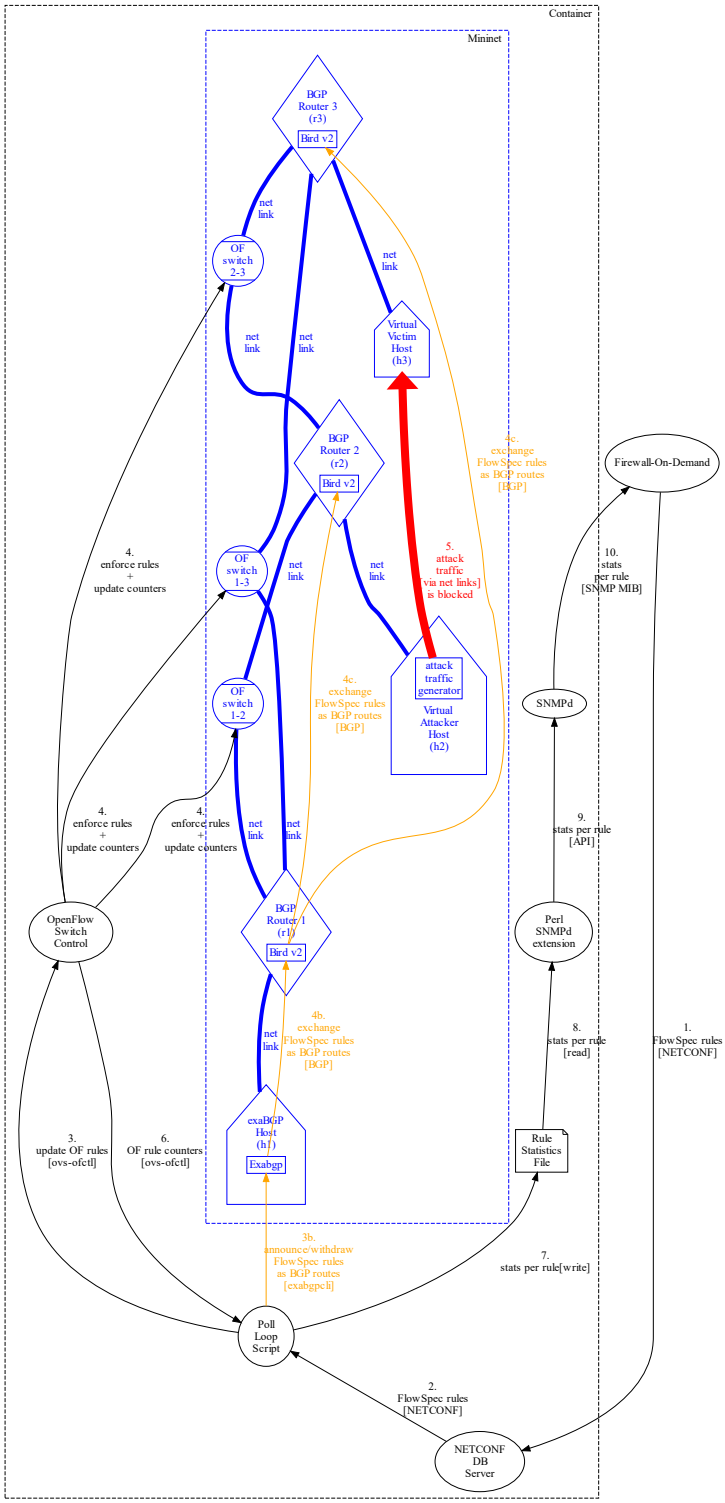

Figure 4.3: Demonstration of FoD mitigation workflow without router hardware (with virtual attacker/victim network and ExabGP and BIRD v2 running inside; BGP involved for exchanging the FlowSpec rules)

Here, the virtual Mininet hosts are segregated in two groups, virtual attacker/victim hosts (as in the case of the vnet2 container) and virtual hosts running the BIRD v2 routing daemon (in that sense, routers). Because of this, the Mininet topology used here plays a more important role than in the vnet2 container. The actual mitigation of packets is not explicitly covered here, as BIRD v2 does not itself support any mitigation. In parallel to the BGP communication aspects, mitigation on OpenFlow switches – as in the case of the vnet2 container – is still performed (steps 1 to 10 in Figure 4.3), but is not triggered by BIRD v2 and so is not related to BGP communication.

Currently, there is one specific example Mininet topology for vnet2-exabgp, actually a so-called Mininet custom topology (for specifying aspects such as the naming of virtual hosts according to the segregation mentioned above – h1, h2, h3 vs. r1, r2, r3 – and for allowing hosts r1, r2, r3 to be in the middle of the Mininet network and have multiple interfaces). The current scripts for starting BIRD v2 on startup of the container are suited for this specific topology. This current Mininet topology, also displayed in Figure 4.3, is basically a triangle of Mininet hosts r1, r2, r2 representing routers. Between each pair of these routers an OpenFlow switch is inserted to also allow mitigation enforcement in this scenario, but which is independent of BGP communication as mentioned above. Each of the r1, r2, r3 hosts has a directly connected proper host: h1, h2, h3, respectively. r1, r2, r3 represent routers and are running BIRD v2 instances, and on the network links in the triangle between them they are communicating via BGP. BGP is used here for exchanging BGP routes as normal, especially for establishing and handling the routing between h1, h2, h3 via the triangle, as well as for exchanging BGP FlowSpec rules. On host h1, ExaBGP is running, which is fed with FlowSpec rules via the NETCONF DB poll loop using exabgpcli's *announce* and *withdraw* commands (step 3a in Figure 4.3). In turn, ExaBGP on h1 has a BGP connection to r1, to share its FlowSpec announcements and withdrawals with it (step 4b). r1 then exchanges these received FlowSpec announcements and withdrawals with its other BGP peers r2 and r3 (step 4c). In Figure 4.3 h2 acts as attacker and h3 as victim.

With this setup, adding or deleting a rule in FoD results in FlowSpec rules being installed in all the BIRD v2 instances on r1, r2, r3. There are commands for querying the state of the current FlowSpec inside ExaBGP as well as each BIRD v2 instance. Therefore, with the vnet2-exabgp container, the BGP communication part of the mitigation workflow can be demonstrated or investigated. Additionally, it shows how router hardware not supporting NETCONF for FlowSpec injection can be connected with FoD, at least for testing purposes.

5 Further Enhancements

Beyond the core service development, the Docker support and the testing on different layers, as described in the previous sections, enhancements were also made in the areas of documentation support and Elasticsearch/Kibana integration.

A pre-existing FoD documentation set was integrated into the web UI and was updated to the current version [GN_FoD_Doc]. Currently, installation and configuration and the use of the REST API is covered. In future, this documentation is to be enhanced and extended to cover the use of the web UI.

Work on an Elasticsearch/Kibana integration was carried out for test and experimentation purposes [Elasticsearch], [Kibana]. Development of the multi-tenant Kibana dashboard for FoD in RENAM was started. Generalisation of Elasticsearch/Kibana Docker configuration and scripts is planned for the future.

6 Conclusion and Outlook

During GN4-3, WP8 T3.4 has developed and tested a range of significant enhancements and improvements to Firewall on Demand. In particular:

- The application was fully ported to Python 3.
- The REST API was enhanced, including for cooperation and coupling with NeMo.
- Support for rules with IPv6 addresses was added.
- Use of Docker containers was introduced.
- OS/systemd installation support was added.
- Automated testing of the web UI and REST API and integration testing with a virtual attacker/victim router/network were added.

Subject to any findings and recommendations arising from the Product Lifecycle Management (PLM) process (FoD is in the "Service Operation (continuous improvement)" stage of the process), piloting of these enhancements and improvements is expected to take place in the second half of 2022, with deployment into production before the end of the year. A more detailed schedule will be defined in due course.

Meanwhile in the near- and long-term future, the following areas are to be worked on:

- Perform the final tests for the REST API enhancements and establish cooperation with the NeMo service.
- Improve and extend Selenium tests.
- Work on pytest to extend it further, e.g., use it for code coverage testing.
- Work on Elasticsearch/Kibana integration.
- Extend internal documentation.

Firewall on Demand continues to be a key component of GÉANT's Network Security Handling and Response Process offering, with the NREN users providing valuable input towards identifying and prioritising areas for development, ensuring that it addresses real needs and remains relevant, useful and used.

References

[Beanstalk]	https://beanstalkd.github.io/
[BIRD]	https://bird.network.cz/
[Celery]	https://www.fullstackpython.com/celery.html
[django-reg]	https://django-registration.readthedocs.io/en/3.3/
[Elasticsearch]	https://www.elastic.co/what-is/elasticsearch
[ExaBGP]	https://github.com/Exa-Networks/exabgp
[GN4-2_D8.12]	GN4-2 Deliverable D8.12 *Distributed Denial of Service Mitigation v1.0 Pilot Follow-up*, 30 November 2018 https://resources.geant.org/wp-content/uploads/2022/07/D8.12_Distributed-Denial-of-Service-Mitigation-v1.0-Pilot-Follow-up.pdf
[GN4-2_D8.2]	GN4-2 Deliverable D8.2 *Firewall on Demand Progress Report*, 12 July 2017 https://resources.geant.org/wp-content/uploads/2022/07/D8.2_Firewall-on-Demand-Progress-Report.pdf
[GN4-2_D8.3]	GN4-2 Deliverable D8.3 *Distributed Denial of Service Mitigation v1.0 Pilot*, 4 September 2017 https://resources.geant.org/wp-content/uploads/2022/07/D8.3_Distributed-Denial-of-Service-Mitigation-v1.0-Pilot.pdf
[GN_FoD_Doc]	https://flowspy.readthedocs.io/en/latest/
[GN_FoD_GitHub]	https://github.com/GEANT/FOD
[GN_FoD_Iface]	https://fod.geant.org [subscription required]
[GN_FoD_Svc]	https://security.geant.org/firewall-on-demand/
[Gunicorn]	https://gunicorn.org/
[hping3]	http://www.hping.org/hping3.html
[iPerf]	https://iperf.fr/
[Kibana]	https://www.elastic.co/what-is/kibana
[M8.4]	Milestone M8.4 *FoD Future Development Requirements Analysis* https://geant.org/wp-content/uploads/2022/07/M8.4_FoD_Future_Development_Requirements_Analysis.pdf
[Mininet]	http://mininet.org/
[NeMo]	https://security.geant.org/nemo-ddos-software/
[netconfd]	https://yuma123.org/wiki/index.php/Yuma_netconfd_Manual
[Netopeer2]	https://github.com/CESNET/Netopeer2
[NSHaRP]	https://network.geant.org/network_operations/
[ping]	https://en.wikipedia.org/wiki/Ping_(networking_utility)
[pytest]	https://docs.pytest.org/en/7.1.x/
[Redis]	https://redis.com/solutions/use-cases/messaging/

[RFC_5575] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch and D. McPherson, *Dissemination of Flow Specification Rules*, RFC 5575, August 2009 https://datatracker.ietf.org/doc/html/rfc5575

[RFC_8955] C. Loibl, S. Hares, R. Raszuk, D. McPherson and M. Bacher, *Dissemination of Flow Specification Rules*, RFC 8955, December 2020

[Selenium] https://www.selenium.dev/

[Shibboleth] https://www.shibboleth.net/

[snmpd] http://www.net-snmp.org/wiki/index.php/Snmpd

[systemd] https://systemd.io/

[tcpdump] https://www.tcpdump.org/

[TShark] https://www.wireshark.org/docs/man-pages/tshark.html

[Warden] https://warden.cesnet.cz/

Glossary

API	Application Programming Interface
BIRD	BIRD Internet Routing Daemon
BGP	Border Gateway Protocol
CERN	European Organisation for Nuclear Research
DB	Database
DDoS	Distributed Denial of Service
FoD	Firewall on Demand
GNU	A recursive acronym for “GNU's Not Unix”
ID	Identifier
IETF	Internet Engineering Task Force
IP	Internet Protocol
MIB	Management Information Base
NeMo	Network Monitoring
NETCONF	Network Configuration Protocol
NOC	Network Operations Centre
NREN	National Research and Education Network
NSHaRP	Network Security Handling and Response Process
PLM	Product Lifecycle Management
REST	Representational State Transfer
RFC	IETF Request for Comments
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
WP8	GN4-3 Work Package 8 Security
WP8 T3	WP8 Task 3 Products and Services
WP8 T3.3	WP8 Task 3 Products and Services: Distributed Denial of Service (DDoS) Mitigation
WP8 T3.4	WP8 Task 3 Products and Services: Firewall on Demand