Abstract:

After having fully designed the DOCTOR architecture in WP1, and after having identified and characterized the main security issues affecting NDN in the task 2.1, this deliverable describes how the global monitoring architecture of the DOCTOR project can be used to secure the operation of a virtualized NDN infrastructure. First, we describe our monitoring architecture, its components and their roles. Second, we add new rules about the NDN and NFV environments to Muval, to make it be able to perform a proactive security analysis based on attack graphs evaluation. Then, we define and evaluate specialized detection algorithms that can detect the main attack scenario we consider against NDN, as described in T2.1 (interest flooding attack, content poisoning attack, mixed NDN/NFV attack, information leakage). The ability of our architecture to detect all the considered attacks now opens the way to the definition of remediation activities to be developed in the WP3.
Deliverable D2.2: Security monitoring of NDN through virtualized components

TABLE OF CONTENTS

1 Introduction ......................................................................................................................................... 3
2 Risk analysis of the DOCTOR architecture based on attack graphs ........................................... 4
  2.1 Introduction to MulVal .................................................................................................................. 4
  2.2 General rules for virtual environments ....................................................................................... 5
  2.3 NDN-specific Rules ...................................................................................................................... 7
  2.4 CyberCaptor processing ............................................................................................................... 10
    2.4.1 Topological data input .......................................................................................................... 10
    2.4.2 Attack graph engine computation result access ................................................................. 11
3 DOCTOR’s global monitoring architecture ................................................................................... 14
  3.1 Monitoring components .............................................................................................................. 14
  3.2 Monitored Metrics ...................................................................................................................... 17
4 Application of the monitoring architecture for NDN security ...................................................... 19
  4.1 Detection of the Interest Flooding Attack (IFA) ........................................................................ 19
    4.1.1 Definitions and assumptions ............................................................................................... 19
    4.1.2 Statistical hypothesis testing theory – the basics ............................................................... 20
    4.1.3 Detection problem statement ............................................................................................... 21
    4.1.4 Detection with known packet-loss rate .............................................................................. 22
    4.1.5 Detection with unknown packet-loss rate .......................................................................... 23
      4.1.5.1 Packet-loss Rate Model ............................................................................................... 23
      4.1.5.2 Proposed Test for unknown packet loss rate ............................................................... 24
    4.1.6 Numerical Results ................................................................................................................. 25
      4.1.6.1 Scenario setup ............................................................................................................... 25
      4.1.6.2 Numerical results on simulated data ............................................................................ 25
      4.1.6.3 Numerical results on ndnSIM data .............................................................................. 26
  4.2 Detection of the Content Poisoning Attack (CPA) ...................................................................... 28
    4.2.1 Principal Component Analysis result and detection metric selection ................................ 28
    4.2.2 Metrics’ distribution ........................................................................................................... 29
    4.2.3 CPA local detector .............................................................................................................. 30
  4.3 Detection of a mixed attack scenario (NFV/NDN) .................................................................... 33
  4.4 Detection of information leakage in NDN ................................................................................. 35
    4.4.1 Problem of Information Leakage in NDN ......................................................................... 35
    4.4.2 Detection with Name-Based Filters ................................................................................... 37
    4.4.3 Performance Evaluation ..................................................................................................... 38
5 Conclusion ......................................................................................................................................... 42
6 References .......................................................................................................................................... 43
1 Introduction

The goal of the DOCTOR project is to provide new assets enabling the secure deployment and operation of NDN. One part consists in running NDN components in a virtualized architecture to take advantage of its flexibility, low cost and ability to be orchestrated and quickly react to events, in order to increase the security, the resilience and the QoS of the whole network architecture. Before being able to react, the first step is however to be able to gather reliable information about the current state of the NDN network, and more precisely to detect the most common attacks. This deliverable shows how the monitoring architecture we designed is able to fulfil this objective.

In the previous task (T2.1) of this work package, which addresses the security topic, we conducted a comprehensive study of the main security vulnerabilities and threats that can affect the DOCTOR virtualized NDN architecture. We showed that NDN is vulnerable to critical attacks that have been lightly covered by the state of the art. On the opposite, NFV offers better security properties by design, inherited from system security through virtualization and isolation. We decided to exhibit and address four specific scenarios considered as the most critical for NDN adoption: three of them are related to each main component of the NDN router (PIT flooding attack, CS poisoning, FIB exploit), while the last one covers both NFV and NDN technologies to highlight the possible bridges that may be exploited by advanced attacks. In this way, we consider security issues that can affect both NDN itself and the way it is virtualized through NFV. For each of those attacks, we carefully described the scenario we consider and we evaluated the impact of each attack. Based on these security threats that have been identified and reproduced, this second task (T2.2) aims to build monitoring functions that can detect these attacks thanks to a proper modelling of their impact on different dimensions (traffic, caches, forwarding tables, etc.), which are evidences that must be caught by our detectors.

Several initiatives already exist to monitor an NDN network. First, the NDN routing demon itself (nfd) includes a monitoring interface [28] gathering per face statistics about the router’s activity. Some research works also address the monitoring of NDN. For instance, [29] proposes a way to test the path taken by a delivered content from the provider to the client. Yet, NDN monitoring is still in an early stage and more advanced and convenient metrics, like a network flow in the IP world, are missing to properly operate a NDN network. When we take into consideration our security objective, the current state of NDN monitoring solutions is not sufficient as the monitoring functions are never coupled with security tools able to detect attacks. Moreover, our specific virtualized NDN architecture leverages NFV technologies, which is also an opportunity to instrument our Virtualized Network Functions (VNF) to give them the ability to be precisely monitored and to design some VNF dedicated to our monitoring and security objectives.

To complete our monitoring architecture, we will also consider a risk-based approach of monitoring. Indeed, the complexity of modern system and networking technologies, with an extensive use of virtualization and encapsulation, make it difficult even for expert human beings to foresee the attack paths that could endanger the whole infrastructure. So we leverage Mulval, whose attack graph models and inference engine will be extended to consider the new identified threats for NDN and new network topologies based on NFV. The resulting knowledge should help to anticipate attacks (proactive approach) to better react to them.

The rest of this document is organized as follow. Section 2 defines new rules about NDN and NFV the risk analysis of the DOCTOR architecture in order to later foresee more complex attacks identified through the analysis of attack graphs. Section 3 describes DOCTOR’s monitoring architecture: what is monitored and how? Section 4 presents the detectors we built for the main attacks previously identified against NDN, plus a new one consisting in information leakage through Interests’ name. Finally, Section 5 concludes this document and introduces our next work for WP3.
2 Risk analysis of the DOCTOR architecture based on attack graphs

2.1 Introduction to MulVal

The DOCTOR project is built upon two key technologies: NFV (Network Virtualization Function) as defined by the European Telecommunications Standards Institute (ETSI) and ICN (Information-Centric Networking) which proposes an Internet data plane that shifts from host-based network mechanisms to content-based ones.

Such an innovative environment, relying on a Virtual Network Function Infrastructure (virtualized machines and virtualized network equipment), and the parallel usage of both an IP and a disruptive ICN network stacks, has a lot of technical specificities that are to be taken into account when analysing security threats.

The analysis of the threats related to the usage of the NDN protocol deployed in a virtualized environment has led to the definition of specific threats that are to be handled in order to lower the security risk involved in introducing such protocol. This would otherwise prevent networks operators from adopting it.

Proactive security enforcement in the DOCTOR project is provided through the usage of the MulVal tool (Multihost, multistage Vulnerability Analysis), embedded into the CyberCAPTOR security monitoring tool. MulVal is a framework for modelling the interaction of software bugs with system and Network configuration. It uses Datalog as its modelling language for the elements in the analysis (bug specification, configuration description, reasoning rules, operating-system permission and privilege model).

Datalog is a declarative logic language in which each formula is a function-free Horn clause (a subset of the Prolog language). A term that begins with a capital letter or _ is a logical variable, the special one _ being used to express that its value is meaningless, otherwise it is a constant.

MulVal knowledge base has two main components:
- A dynamic part that describes on one hand the hosts and network configuration, and on the other hand the vulnerability database,
- A static part that describes the rules driving the interactions between the various system components.

The dynamic part is being generated from a vulnerability database and from the provided topological description by the CyberCAPTOR system. Both parts are then merged and fed into MulVal in order to trigger the construction of the attack graphs relative to the described environment.

Therefore, the static part of the Mulval rules had to be updated in order to cope with the complexity of the DOCTOR environment, and the dynamic part production modified in order to include topological information in the NFV and NDN frame.
2.2 General rules for virtual environments

The first Mulval rules set evolution consists in being able to describe the mixed physical and virtual environment, by instance the relationship between a VM and its physical host and between the VM and its orchestrator. Then rules have been extended to introduce Network Function Virtualization (NFV) capabilities. The third step was to introduce specificities of the Named Data Networking (NDN) architecture implementation, topology and identified specific threats.

New kind of security impact have been introduced in rules, beside “remote code execution”, “privilege escalation”, specific NDN actions “output compromised signed” and “output compromised unsigned” were created as result of attacks capable of altering remotely a NDN router output.

Rules were also added, in order to add NDN specific attacks (CPA, Malicious RA, Interception).

Some rules extracted from the CyberCAPTOR rules set for Mulval are explained hereunder in order to illustrate some important parts of the rules set.

**primitive predicates declaration**

primitive(vmOnHost(_vm,_host,_software,_user)).

The rule defines the properties attached to a VM execution that is the physical machine it is running on, the executed software and the user that runs it.

primitive(vmInDomain(_vm,_orchestrator)).

The rule is used to link the virtual machines to the orchestrator that manage them.

primitive(vnfOnPath(_vnf,_host1,_host2,_port,_daemon,_user)).

This rule defines a VNF path location.

primitive(localServiceInfo(_servicename, _host, _program, _user)).

This rule is used to define local service information. It is defined by its name, the host it is running on, the software and the user that runs it.

**derived predicates declaration**

derived(execCode(_host,_user)).

This rule defines the capability of a user to execute some code on a host.

derived(accessFile(_machine,_access,_filepath)).

This rule defines the capability of accessing some part of a machine filesystem.

derived(orchestratorCompromised(_orchestrator)).

This rule defines the fact that an orchestrator has been compromised.

**interaction rules**

interaction_rule(
    (execCode(Hypervisor, User) :-

Deliverable D2.2: Security monitoring of NDN through virtualized components

execCode(Vm, _),
vmOnHost(Vm, Hypervisor, Software, User),
vulExists(Hypervisor, Vuln, Software),
vulProperty(Vuln, localExploit, privEscalation)),
rule_desc('can escape from VM to host', l)).

interaction_rule(
  (execCode(Hypervisor, User) :-
   execCode(Vm, _),
   vmOnHost(Vm, Hypervisor, Software, User),
   vulExists(Hypervisor, Vuln, Software, localExploit, privEscalation)),
rule_desc('can escape from VM to host', l)).

These two rules express some conditions under which a user may be capable of running code on
an hypervisor host, that is the hypervisor software has a vulnerability that permits a user able to run
code of the virtual machine, to perform a local exploit resulting in a privilege escalation leading to
code execution on the physical host.

interaction_rule(
  (execCode(Host, User) :-
   execCode(Host, root), User \= root),
rule_desc('root can impersonate any user', l)).

This rule expresses that the root user is able to impersonate any other user on a local machine.

interaction_rule(
  (execCode(Vm, root) :-
   vmOnHost(Vm, Hypervisor, _, User),
   execCode(Hypervisor, User)),
rule_desc('VM runs on compromised host so it is compromised', l)).

This rule expresses that a user able to execute code on the hypervisor host, may run code as root
on a local virtual machine.

interaction_rule(
  (accessFile(Vm, _, _) :-
   vmOnHost(Vm, Hypervisor, _, User),
   execCode(Hypervisor, User)),
rule_desc('Can access VM filesystem through hypervisor', l)).

This rule expresses that a user able to execute code on the hypervisor host, may ACCESS a local
virtual machine file system.

interaction_rule(
  (orchestratorCompromised(Orchestrator) :-
   localServiceInfo(Orchestrator, Host, Program, User),
   execCode(Host, User)),
rule_desc('Can take over orchestrator from its host', l)).

This rule expresses that a user able to execute code on the orchestrator host, may compromise an
orchestrator running on this host.

interaction_rule(
  (execCode(Vm, root) :-
   vmInDomain(Vm, Orchestrator),
   orchestratorCompromised(Orchestrator)),
rule_desc('Host controlled by compromised orchestrator or controller', l)).

This rule expresses that a user on a compromised orchestrator host, may execute code on the vir-
tual machines of its domain.

interaction_rule(
  (execCode(Vnf, User) :-
   vnfOnPath(Vnf, Host1, Host2, Port, Software, User),
   execCode(Host1, _)),
rule_desc('Can execute code through orchestrator', l)).
Deliverable D2.2: Security monitoring of NDN through virtualized components

This rule expresses the conditions by which a user can execute code on a VNF, when vulnerability exists on this Vnf software, and that a user can execute code on an host that is on its path.

### 2.3 NDN-specific Rules

#### primitive predicates declaration

- primitive(hasNDNFace(_host, _face)).
- primitive(faceIsLinked(_faceA, _faceB)).
- primitive(isNDNRouter(_host)).
- primitive(isIGWSoftware(_software)).
- primitive(ndnServiceInfo(_host, _software, _user)).

These rules define the properties attached to NDN, that is the association between a host and a face, existence of a link between two faces, characterization of a host machine as a NDN router, characterization of a software as a NDN Gateway, and definition of the information relative to a NDN service info.

#### derived predicates declaration

- derived(ndnLink(_host1, _host2)).

This rule defines that two hosts have a NDN link between them.

- derived(ndnOutputCompromisedSigned(_ndnRouter)).
- derived(ndnOutputCompromisedUnsigned(_ndnRouter)).
- derived(ndnTrafficIntercepted(_ndnRouter)).

These rules define possible NDN specific security impacts.

#### interaction rules

- interaction_rule(
  (ndnLink(Host1, Host2) :-
  hasNDNFace(Host1, Face1),
  hasNDNFace(Host2, Face2),
  faceIsLinked(Face1, Face2)),
rule_desc('Direct NDN link exists between hosts', 1.0)).

- interaction_rule(
  (ndnLink(Host1, Host2) :-
  hasNDNFace(Host1, Face1),
  hasNDNFace(Host2, Face2),
  faceIsLinked(Face2, Face1)),
rule_desc('Direct NDN link exists between hosts', 1.0)).

These rules compute the NDN topology by creating NDN Link from Faces, in both directions, assuming NDN links are symmetrical.

- interaction_rule(
  (ndnOutputCompromisedSigned(NDNRouter1) :-
  ndnLink(NDNRouter1, NDNRouter2),
  isNDNRouter(NDNRouter1),
  ndnOutputCompromisedSigned(NDNRouter2)),
rule_desc('NDN router transfers signed but compromised data', 1.0)).
Deliverable D2.2: Security monitoring of NDN through virtualized components

This rule expresses the case of a compromised and signed input, leading to an output compromised and signed, therefore a Cache Poisoning with fake data attack.

```
interaction_rule(
  {ndnOutputCompromisedSigned(NDNRouter1) :-
    ndnLink(NDNRouter1, NDNRouter2),
    vulExists(NDNRouter1, Vuln, Software),
    vulProperty(Vuln, signatureExploit, cachePoisoned),
    rule_desc('NDN router can transfer unsigned and compromised data', 1.0)).
```

This rule expresses the case of a compromised and unsigned input, leading to an output compromised and unsigned, therefore a Cache Poisoning with corrupted data.

```
interaction_rule(
  {execCode(NDNRouter1, User) :-
    ndnLink(NDNRouter1, NDNRouter2),
    ndnServiceInfo(NDNRouter1, Software, User),
    vulExists(NDNRouter1, Vuln, Software),
    vulProperty(Vuln, remoteExploit, privEscalation),
    rule_desc('Can execute code on router', 1.0)).
```

This rule expresses the case of a compromised and signed input, beside a vulnerability that permits privilege escalation, leading to an output compromised and unsigned, therefore a Cache Poisoning with fake data.

```
interaction_rule(
  {execCode(NDNRouter1, User) :-
    ndnLink(NDNRouter1, NDNRouter2),
    ndnOutputCompromisedUnsigned(NDNRouter2),
    ndnServiceInfo(NDNRouter1, Software, User),
    vulExists(NDNRouter1, Vuln, Software),
    vulProperty(Vuln, remoteExploit, privEscalation),
    rule_desc('Can execute code on router', 1.0)).
```

This rule expresses the case of a compromised and unsigned input, beside a vulnerability that permits privilege escalation, leading to an output compromised and unsigned, therefore a Cache Poisoning with corrupted data.

```
interaction_rule(
  {execCode(NDNRouter1, User) :-
    ndnLink(NDNRouter1, NDNRouter2),
    ndnOutputCompromisedUnsigned(NDNRouter2),
    ndnServiceInfo(NDNRouter1, Software, User),
    vulExists(NDNRouter1, Vuln, Software),
    vulProperty(Vuln, signatureExploit, cachePoisoned),
    rule_desc('NDN router can transfer unsigned and compromised data', 1.0)).
```

This rule expresses that when code execution is possible, this can lead to a compromised and signed output, therefore production of fake data.

```
interaction_rule(
  {execCode(NDNRouter1, User) :-
    ndnOutputCompromisedUnsigned(Host),
    rule_desc('Attacker can modify data sent by host or router', 1.0)).
```

This rule expresses that when code execution is possible, this can lead to a compromised and unsigned output, therefore production of corrupted data.

```
interaction_rule(
  {execCode(NDNRouter1, User) :-
    ndnOutputCompromisedSigned(Host),
    rule_desc('Attacker can modify data sent by host or router', 1.0)).
```

This rule expresses that when code execution is possible, this can lead to a compromised and unsigned output, therefore production of corrupted data.

```
interaction_rule(
  {execCode(Host, User) :-
    hasIP(Host, IP),
    netAccess(IP, Protocol, Port),
    networkServiceInfo(IP, Software, Protocol, Port, _),
    isIGWSoftware(Software),
    rule_desc('Attacker can modify data sent by host or router', 1.0)).
```
Deliverable D2.2: Security monitoring of NDN through virtualized components

```
rule_desc('Attacker can cross IGW from IP to NDN domain', 1.0)).

interaction_rule(  
  (ndnOutputCompromisedUnsigned(Host) :-  
    hasIP(Host, IP),  
    netAccess(IP, Protocol, Port),  
    networkServiceInfo(IP, Software, Protocol, Port, _),  
    isIGWSoftware(Software),  
    rule_desc('Attacker can cross IGW from IP to NDN domain', 1.0)).

These rules expresses that a malicious packet may come from the IP domain to the NDN domain using the Internal Gateway.

interaction_rule(  
  (netAccess(IP, Protocol, Port) :-  
    hasIP(Host, IPGW),  
    hacl(IPGW, IP, Protocol, Port),  
    ndnServiceInfo(Host, Software, _),  
    ndnLink(Host, Host2),  
    ndnOutputCompromisedSigned(Host2)),  
  rule_desc('Attacker can cross EGW from NDN to IP domain', 1.0)).

These rules expresses that a malicious packet may come from the NDN domain to the IP domain using the External Gateway.

interaction_rule(  
  (ndnOutputCompromisedSigned(Host) :-  
    ndnLink(Host, Host2),  
    execCode(Host2, _),  
    isNDNRouter(Host),  
    vulExists(Host, Vuln, Software),  
    vulProperty(Vuln, pitExploit, cachePoisonned)),  
  rule_desc('Attacker can send replies to interests issued to another interface', 1.0)).

interaction_rule(  
  (ndnOutputCompromisedUnsigned(Host) :-  
    ndnLink(Host, Host2),  
    execCode(Host2, _),  
    isNDNRouter(Host),  
    vulExists(Host, Vuln, Software),  
    vulProperty(Vuln, pitExploit, cachePoisonned)),  
  rule_desc('Attacker can send replies to interests issued to another interface', 1.0)).

These rules expresses that a router on which code can be ran and that has vulnerability, may reply to interest from another face.

interaction_rule(  
  (ndnTrafficIntercepted(Host) :-  
    ndnLink(Host, Host2),  
    execCode(Host2, _),  
    isNDNRouter(Host),  
    ndnTrafficIntercepted(Host)),  
  rule_desc('Attacker can send replies to interests issued to another interface', 1.0)).
```
Deliverable D2.2: Security monitoring of NDN through virtualized components

vulExists(Host, Vuln, Software),
vulProperty(Vuln, fibExploit, corruptFib)),
rule_desc('Attacker can redirect router upstream to controlled router to intercept traffic', 1.0)).

This rule expresses a vulnerability of a router accepting FIB announcements without signature veri-

2.4 CyberCaptor processing

2.4.1 Topological data input

CyberCAPTOR is composed of two sub-components: cybercaptcha-server which is the attack graph computation engine and cyber-data-extract which is in charge or interfacing cybercaptcha-server with external systems:

- access to the external service supplying the up to date topological information
- data conversion to the CyberCAPTOR native format
- cybercaptcha-server initialization REST API invocation, triggering the attack graph recomputation

In the context of the Doctor project, the topological information is provided by the MMT application, using XML formatted messages. Figure 1 is an example of such a topology description message:

```
<machine>
  <name>host1</name>
  <security_requirement>1</security_requirement>
  <controllers>
    <controller>orchestrateur_global</controller>
  </controllers>
  <interfaces>
    <interface>
      <name>eth0</name>
      <ipaddress>192.168.1.1</ipaddress>
      <vlan>
        <name>vlan0</name>
        <label>vlan0</label>
      </vlan>
    </interface>
  </interfaces>
  <services>
    <service>
      <name>kvm</name>
      <ipaddress>192.168.1.1</ipaddress>
      <protocol>none</protocol>
      <port>0</port>
      <global_name>host1_kvm</global_name>
      <cpe>cpe:/a:redhat:kvm:83</cpe>
    </service>
  </services>
  <routes>
    <route>
      <destination>0.0.0.0</destination>
      <mask>0.0.0.0</mask>
      <gateway>192.168.1.254</gateway>
      <interface>eth0</interface>
    </route>
  </routes>
</machine>
```

Figure 1: Example of CyberCaptor’s topological information model
2.4.2 Attack graph engine computation result access

The supplied topological information is then converted to the Datalog format (Figure 2) inside the cybercaptor-server component in order to build the Mulval knowledge base dynamic part.

```
/** Add Host egw */
.getSeconds()
attackerLocated('egw').
attackGoal(execCode('egw', _)).
isNDNRouter('egw').
vmOnHost('egw', 'host4', 'kvm', 'root').
vmInDomain('egw', 'orchestrateur_global').
hasIP('egw', '10.0.3.1').
hasNDFace('egw', 'egw_r6').
hasNDFace('egw', 'egw-ndn4').
hasIP('egw', '10.0.3.1').
vmInVlan('10.0.3.1', 'vlan3').
hostAllowAccessToAllIP('egw').
```

These rules are combined with the Mulval knowledge base static part, which is composed of the rules described in the first part of the chapter. The Mulval tool is then able to infer the possible action of an attacker.

Upon computation of the consequent attack graph and attack paths, the result is made available under the form of a AND-OR tree that describes the steps in which attacks can arise. This tree combines the facts that are asserted from the topological description inputs (stated as Leaf nodes), the detected vulnerabilities (stated as Vulnerability nodes) and the applied rules (stated as And nodes).

This result can be accessed in two different ways. First, the cybercaptor-server component includes a web application (CYBERCAPTOR client) presenting the results under a graphical form in a web browser as illustrated in Figure 3.
Figure 3: Visual representation of an attack graph

Second, the cybercaptor-server REST API provides an access to the computation results in a JSON encoded format (see Figure 4) at the following URLs:

- **Get the attack graph:**
  ```http://cybercaptor-server-base:8080/cybercaptor-server/rest/json/attack_graph```

- **Get the attack paths number:**
  ```http://localhost:8080/cybercaptor-server/rest/json/attack_path/number```

- **Get an attack path using its 0-based index:**
  ```http://localhost:8080/cybercaptor-server/rest/json/attack_path/{i}```

- **Get an attack path remediations using its 0-based index:**
  ```http://localhost:8080/cybercaptor-server/rest/json/attack_path/{i}/remediations```
Deliverable D2.2: Security monitoring of NDN through virtualized components

![JSON representation of an attack graph](image-url)

**Figure 4**: JSON representation of an attack graph
3 DOCTOR’s global monitoring architecture

In the first work package (D1.2), we defined the DOCTOR’s architecture by following ETSI guidelines for building a NFV architecture (see Figure 5), while in the fourth work package (D4.1) we described our early testbed implementation including monitoring components. In this section, we remind how the monitoring architecture of DOCTOR is designed and go deeper into the monitored metrics.

Figure 5: Overview of the DOCTOR virtualized network infrastructure

3.1 Monitoring components

First, our monitoring architecture follows the usual separation between the data plane, which will carry out users’ IP or NDN traffic, from the control or management plane, which will transport the information about the network and system states. Because we operate a virtualized infrastructure, this separation is not physical but enforced thanks to different VLANs, as illustrated in Figure 6.
Then, the monitoring solution in the Doctor project can be divided into three main sectors:

- Local monitoring at the virtual node level: The monitoring probes are located in the node with virtual network function VNF. They allow capturing different local metrics (at the network, function or system levels) and analysing them to detect local security flaws.
- Global monitoring at the tenant level: The monitoring probe allows correlating different metrics provided from local probes to be able to detect complex attacks impacting different network nodes.
- Operator level: A dashboard can be provided to the virtual network operator to allow him to have a global view on the network statistics and security incidents before applying corresponding counter-measures.

In the context of DOCTOR project, the monitoring functionality is provided by the means of the MMT tool (stands for Montimage Monitoring Tool). The local monitoring is based on the MMT SDK that has a plugin architecture to parse different sources of information (e.g., NDN network packets, NFD logs, etc.) and extract the relevant metrics. Both local and tenant based analysis rely on MMT security libraries that allows to correlate computed metrics and detect security incident either locally or at the tenant level. The operator dashboard is an extension of MMT-Operator to take into account NDN characteristics. Figure 7 presents a high level view of the main features of each monitoring component of the architecture, and Figure 8 presents at a lower level how the flow of information is implemented between the components.
Deliverable D2.2: Security monitoring of NDN through virtualized components

Figure 7: Main features of MMT components

Figure 8: DOCTOR’s global monitoring architecture information flow based on MMT tools
3.2 Monitored Metrics

The extraction of metrics in DOCTOR project relies on MMT-SDK that allows extracting different network/function/system events and metrics according to the user needs. It has a modular architecture based on plugins to allow the support of additional protocols and applications, or to support new metrics specified as attributes to be extracted by the plugins. A focus on the NFD Log plugin will be done in this section.

This plugin intends to monitor different metrics at the NDN node level to characterize any change in the node behavior. This behavior change is evaluated at the arrival of each NDN packet which is processed by NDN internal components: Content Store (CS), Pending Interest Table (PIT) and Forwarding Information Base (FIB). Thus, we divide metrics into two categories:

- The external metrics represents the number of packets (Interest/Data/NACK) received and sent by the NDN node. These metrics allow not only to show the characteristics (volume, frequency, the ratio of conversation Interest/Data ...) of traffic but also to compare with the internal metrics, which represents the number of events occurring inside the node and which cannot be observed from outside of the node.

- The internal metrics: In the processing of Interest and Data packets in NDN, an Interest or Data packet is processed by CS, PIT, and FIB. Firstly, an incoming Interest can hit or miss a cache in CS and an incoming Data could be inserted into the CS, these numbers represent the characteristics of CS with current traffic, hence, the monitoring is required. For example in case of CPA, when the attack happens, the Interest from the legitimate user will hit the content poisoned by the attacker, then, CS Hit will increase. Regarding the PIT: an incoming Interest can create, update an entry in the PIT or an incoming Data can remove an entry in PIT. These numbers, as well as the time that an entity (Interest) stays in PIT, and the number of entries in PIT compared to the external metrics, help us to determine how well the current traffic is. For example, if Incoming Interests tend to create more new entries in the PIT than usually, this means that the current traffic is potentially an IFA attack. Finally, in the packet processing, sometimes, the packet is dropped for different reasons, such as in case of a CPA. Indeed, when an attacker succeeds to send fake Data before the good provider, the Data sent by the good provider will be dropped by the NDN router and considered as a Unsolicited Data. Therefore the number of packets dropped is also taken into account. In this process, a normal Interest/Data packet has no impact on the FIB because it is controlled by NFD or its management protocol, hence, it is not modified by packet processing in the data plane in general.

The following table syntheses a list of metrics and their description. Those metrics are correlated and processed with powerful statistical tools in the next section to build detectors of NDN attacks such as IFA or CPA.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Interest</td>
<td>Number of incoming Interest packets</td>
</tr>
<tr>
<td>In Data</td>
<td>Number of incoming Data packets</td>
</tr>
<tr>
<td>In NACK</td>
<td>Number of incoming NACK packets</td>
</tr>
<tr>
<td>Out Interest</td>
<td>Number of outgoing Interest packets</td>
</tr>
<tr>
<td>Out Data</td>
<td>Number of outgoing Data packets</td>
</tr>
<tr>
<td>Out NACK</td>
<td>Number of outgoing NACK packets</td>
</tr>
<tr>
<td>CS Insert</td>
<td>Number of insert in CS</td>
</tr>
<tr>
<td>CS Miss</td>
<td>Number of Cache miss in CS</td>
</tr>
<tr>
<td>PIT Create</td>
<td>Number entries created in PIT</td>
</tr>
<tr>
<td>PIT Update</td>
<td>Number entries updated in PIT</td>
</tr>
</tbody>
</table>
Deliverable D2.2: Security monitoring of NDN through virtualized components

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIT Delete</td>
<td>Number entries deleted in PIT</td>
</tr>
<tr>
<td>PIT Unsatisfied</td>
<td>Number entries unsatisfied in PIT</td>
</tr>
<tr>
<td>PIT Number</td>
<td>Number entries in PIT</td>
</tr>
<tr>
<td>PIT Exist Time</td>
<td>Average of existing time in PIT</td>
</tr>
<tr>
<td>Drop Interest</td>
<td>Number of Interest is dropped</td>
</tr>
<tr>
<td>Drop Data</td>
<td>Number of Data is dropped</td>
</tr>
<tr>
<td>Drop NACK</td>
<td>Number of NACK is dropped</td>
</tr>
<tr>
<td>In Interest</td>
<td>Number of incoming Interest</td>
</tr>
<tr>
<td>Out Data</td>
<td>Number of outgoing Data</td>
</tr>
</tbody>
</table>

The NFD Management Protocol enables collecting data related to the status of an NDN node, e.g. In Interest, PIT Number. However, regarding the metric list that we need to collect, these statistics are not sufficient. To the best of our knowledge, currently, there is no mechanism to collect more complexe metrics that are unavailable in NFD Management Protocol, such as CS Hit, CS Miss, PIT Exist Time. Since NFD is still under heavy development, we chose to avoid modifying its implementation, and instead, we extract all necessary information directly from the log of NFD, what appears to be more sustainable on the long run. For this purpose, a plugin for the MMT probe has been developed and installed in all NDN routers’ containers to extract metrics that we need.

The following figure illustrates an example from the NFD log trace. Each log line corresponds to an event in NFD, including: (1) the timestamp; (2) the event name; (3) the face and (4) the corresponding Interest. Besides, implicit metrics can be deduced from the log. In this example, the logged events (i.e. OnIncomingInterest, OnContentStoreMiss, onOutgoingInterest) provide information to directly update In Interest, CS Miss and Out Interest metrics. In addition, although the creation a PIT entry is not explicitly logged, we can still deduce it since the incoming Interest is not found in the cache (i.e. miss) and is forwarded by NFD, hence leading to an update for the PIT Create metric.

```
1503332255.605719 DEBUG: [Forwarder] onIncomingInterest face=264 interest=/com/good/content4
1503332255.605777 DEBUG: [Forwarder] onContentStoreMiss interest=/com/good/content4
1503332255.605933 DEBUG: [Forwarder] onOutgoingInterest face=265 interest=/com/good/content4
```

**Figure 9. Example of NDF log**

Besides the NFD LOG Plugin, we also implemented an NDN Plugin which allows extracting information from NDN packets. These two plugins are integrated into MMT SDK and then used by MMT-Probe to extract monitoring data from both NDN traffic and NFD Log files. Thus, these metrics are published to a local event bus (see Figure 8) which is subscribed by a local and a tenant MMT Security. The local MMT Security deals with the detection of attacks which could be detected locally like IFA, while the tenant level MMT-Security is deployed in Tenant Controller which take the responsibility to detect attacks in the whole network. By using rules written in XML, the tenant MMT-Security will detect the attack and send alerts to MMT Operator via a global event bus, which is available for all tenants.
4 Application of the monitoring architecture for NDN security

4.1 Detection of the Interest Flooding Attack (IFA)

4.1.1 Definitions and assumptions

For each face of a NDN router, at instant $t^{th}$, the number of incoming Interest packets and outgoing Data packets, denoted as $i_t$ and $d_t$ respectively, will be measured. Ideally, in NDN each incoming Interest at a face should be resolved by one Data packet. However, in any type of networks, a part of the packets could be lost. Hence, let $\ell_t = 1 - d_t/i_t$ be the packet-loss rate measured at instant $t$, i.e. the measured ratio of unresolved Interests. The number of Interests $i_t$ is drawn from a Poisson distribution - a usual model to represent the users’ behavior over the Internet [1]. In addition, following the model proposed in [2], [3], it is assumed that, at instant $t$, all Interests have the same probability of not being resolved, denoted as $p_t$. Therefore, under normal situation, such probability should corresponds to the expectation of the measured packet-loss rate:

$$\mathbb{E}(\ell_t) = p_t.$$ 

As a result, the number of Data packets received $d_t$ follows a binomial distribution:

$$d_t \sim \mathcal{B}(i_t, 1 - p_t),$$

with expectation $\mathbb{E}(d_t) = i_t(1 - p_t)$.

For generality, it is assumed in our work that the packet-loss rate $p$ is unknown, probably changes over time and is measured by $\ell$. However, under normal condition, its fluctuation should not be abrupt. We use this fact to build a model for the packet-loss rate of legitimate traffic. By contrast, when an IFA is started, a significant number of Interests for non-existing content are sent, resulting in an sudden increase of the measured packet-loss rate $\ell$.

Let us denote $N_a$ the number of malicious Interests sent during the IFA by attacker-controlled hosts besides the legitimate Interests (denoted as $i_t^*$). Therefore, the IFA can be characterized by an increase in the number of incoming Interests:

$$i_t = i_t^* + N_a.$$  \hspace{1cm} (1)

It is important to note that distinguishing legitimate Interests $i_t^*$ from the whole flow of Interest packets $i_t$ is not possible. Moreover, $N_a$ additional Interests cannot be resolved, hence increasing the expectation of the measured packet-loss rate:

$$\mathbb{E}(\ell_t) - p_t = a > 0.$$

Another fact should be noticed is that whether an IFA is currently happening or not, the expected number of Data packets received at a given face should remain the same, and mainly depend on the number of legitimate Interests $i_t^*$:

$$(1 - p_t)i_t^* = \mathbb{E}(d_t) = (1 - p_t - a)(i_t^* + N_a)$$

$$\iff \frac{(1 - p_t)N_a}{i_t^* + N_a} = a.$$ 

Hence:

$$a = \mathbb{E}(\ell_t) - p_t = \frac{(1-p_t)N_a}{i_t^* + N_a}.$$ \hspace{1cm} (2)
Table 2: Notations and symbols

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_t$</td>
<td>Number of incoming Interests at time $t^{th}$</td>
</tr>
<tr>
<td>$d_t$</td>
<td>Number of outgoing Data at time $t^{th}$</td>
</tr>
<tr>
<td>$l_t$</td>
<td>Packet-loss rate measured at time $t^{th}$</td>
</tr>
<tr>
<td>$p_t$</td>
<td>Actual packet-loss rate at time $t^{th}$</td>
</tr>
<tr>
<td>$\mathcal{H}_0$</td>
<td>Null hypothesis</td>
</tr>
<tr>
<td>$\mathcal{H}_1$</td>
<td>Alternative hypothesis</td>
</tr>
<tr>
<td>PFA, $\alpha$</td>
<td>Probability of False Alarm</td>
</tr>
<tr>
<td>$\alpha_0$</td>
<td>Prescribed PFA</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Detection power</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Detector’s threshold</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Statistical test</td>
</tr>
<tr>
<td>$\bar{\delta}$</td>
<td>Uniformly Most Powerful test</td>
</tr>
<tr>
<td>$\chi^2$</td>
<td>X converge in distribution to Y</td>
</tr>
<tr>
<td>$P_i(E)$</td>
<td>Probability of event $E$ under hypothesis $\mathcal{H}_i$</td>
</tr>
<tr>
<td>$a$</td>
<td>The increased amount of packet loss rate</td>
</tr>
<tr>
<td>$N_0$</td>
<td>Number of malicious Interests</td>
</tr>
<tr>
<td>$i_t^*$</td>
<td>Number of legitimate Interests</td>
</tr>
<tr>
<td>$N$</td>
<td>Sample size</td>
</tr>
<tr>
<td>$Z$</td>
<td>Space of test’s input</td>
</tr>
</tbody>
</table>

4.1.2 Statistical hypothesis testing theory – the basics

The method we used to design our detection is based on statistical hypothesis testing theory with Neyman-Pearson two-criteria approach since it can provide a consistent most powerful test that does not depend on router’s characteristics or measured values. Besides, this statistical approach allows establishing false-alarm and setting up a threshold such that the prescribed performance can be ensured. Moreover, this method allows us to compare the empirical performance of our test with the theoretically established one, rendering the proposed test well-grounded and more reliable.

The input of hypothesis testing is a sample $Z_N$, $Z_N \in Z$. This sample is a set of $N$ empirical realizations of a random variable $z$. A statistical hypothesis $\mathcal{H}_j$ refers to a set of parameters vectors $\theta_j$. Each vector $\theta$ in this set defines a possible probability distribution $\mathbb{P}_\theta$ of $Z_N$ [4]:

$$\mathcal{H}_j = \{ Z_N \sim \mathbb{P}_\theta, \theta \in \Theta_j \}.$$ 

A hypothesis $\mathcal{H}_j$ is called simple when there is only one unique $\theta$ in $\Theta_j$. On the contrary, it is called composite. In the usual case of binary statistical tests, there are two hypotheses: (1) null hypothesis $\mathcal{H}_0$ and (2) alternative hypothesis $\mathcal{H}_1$. $\mathcal{H}_0$ is usually the normal case and $\mathcal{H}_1$ is usually the abnormal case that we want to detect. A statistical test $\delta$ between two hypotheses $\mathcal{H}_0, \mathcal{H}_1$ is a subjective and measurable mapping from the sample space $Z$ to the set of hypotheses [4]:

$$\delta: Z \rightarrow \{ \mathcal{H}_0, \mathcal{H}_1 \}.$$ 

In order to design a good statistical test with the Neyman-Pearson approach, there are some key concepts which should be aware of: (1) probability of false alarm, (2) detection power, (3) Likelihood Ratio and (4) the uniformly most powerful test. The set $\Theta_0$ defining $\mathcal{H}_0$ contains many parameters $\theta_0$. For each of these parameters, there is a probability that the test $\delta$ rejects the null hypothesis $\mathcal{H}_0$ while it is actually true. The greatest value of these probabilities is called the Probability of False Alarm (PFA) of the test $\delta$, denoted by $\alpha(\delta)$ [4]. Meanwhile, the Detection Power of a test $\delta$, for a parameter $\theta_1 \in \Theta_1$, is the probability that $\mathcal{H}_1$ is detected correctly, denoted by $\beta(\theta_1, \delta)$ [4]:
\[
\alpha(\delta) = \sup_{\theta \in \Theta} \mathbb{P}_{\theta}[\delta(Z_N) = \mathcal{H}_1], \\
\beta(\theta_1, \delta) = \mathbb{P}_{\theta_1}[\delta(Z_N) = \mathcal{H}_1].
\]

For a prescribed false alarm probability \( \alpha_0 \), we define the class of test \( \mathcal{K}_\alpha \) containing all the tests whose false alarm probability is lower than \( \alpha_0 \):

\[ \mathcal{K}_\alpha = \{ \delta : \alpha(\delta) \leq \alpha_0 \}. \]

A Uniformly Most Powerful (UMP) test \( \hat{\delta} \) in the class \( \mathcal{K}_\alpha \) is a test providing the highest power under all the parameters \( \theta_1 \in \Theta_1 \) [4]:

\[ \forall \delta \in \mathcal{K}_\alpha, \quad \forall \theta_1 \in \Theta_1, \quad \beta(\theta_1; \delta) \leq \beta(\theta_1; \hat{\delta}). \]

For simple hypotheses, since \( \theta \) is unique for each hypothesis, the test \( \hat{\delta} \) is called the Most Powerful (MP) test.

The idea of Neyman-Pearson two-criteria approach is to design a test in the class \( \mathcal{K}_\alpha \) that can warrant a pre-defined false alarm probability \( \alpha \) and maximizes the test power \( \beta(\theta_1, \delta) \). In the case of simple hypotheses, according to the Neyman-Pearson lemma [4], the most powerful test \( \hat{\delta} \) is the Likelihood Ratio (LR) test:

\[ \hat{\delta}(Z_N) = \begin{cases} 
\mathcal{H}_0, \text{ if } \Lambda(Z_N) = \frac{f_1(Z_N)}{f_0(Z_N)} < \tau \\
\mathcal{H}_1, \text{ if } \Lambda(Z_N) \geq \tau 
\end{cases} \]

in which \( \Lambda(Z_N) \) is the LR and \( f_j \) is the probability density of \( \mathbb{P}_j, j = 0,1 \). LR test can be transformed by applying a monotone function to both side of the inequality in (1). The threshold \( \tau \) is the solution of the equation:

\[ \mathbb{P}_0[\Lambda(Z_N) \geq \tau] = \alpha_0. \]

Meanwhile, in the case of composite hypotheses, the UMP test barely exists in reality. Fortunately, the testing theory for this type of hypotheses has been well-developed for some particular cases. Due to the space constraint, only cases that are used to achieve the results will be presented in corresponding sections.

### 4.1.3 Detection problem statement

As mentioned in the previous section, the hypothesis \( \mathcal{H}_0 \) is usually the normal case and \( \mathcal{H}_1 \) is usually the abnormal case that we want to detect. Therefore, according to IFA’s description in the deliverable D2.1, the detection problem against IFA consists in choosing between two hypotheses:

- \( \mathcal{H}_0 \): “the number of Interests sent \( i_t \) and Data packets received \( d_t \) are consistent with what is expected from \( p_t \), i.e. there is no occuring IFA”
- \( \mathcal{H}_1 \): “the number of Interest packets sent \( i_t \) is significantly higher than what is expected from \( d_t \) and \( p_t \), i.e. an IFA is occuring”

Those two hypotheses can be rewritten formally as:

\[
\begin{align*}
\mathcal{H}_0 &: d_t \sim B \left( i_t, 1 - p_t \right), \\
\mathcal{H}_1 &: d_t \sim B \left( i_t - N_a, 1 - p_t \right), N_a > 0. 
\end{align*}
\]

The hypotheses as formulated in the equation above highlight the main difficulties of the present testing problem. First, we note that the IFA implies a change on both expectation and variance of the measured loss-packet rate \( \ell_t \). Second, the parameters of attack payload \( N_a \) or \( a \) (Equations \( i_t = i_t^* + N_a \)). Third, the undoubtedly greatest difficulty is that the expected loss-packet rate is unknown in practice.
4.1.4 Detection with known packet-loss rate

In this section, we first address the case when the expected packet-loss rate $p_t$ is known. As a result, this allows designing the theoretical optimal Likelihood Ratio Test (LRT) and also permits the assessment of its statistical performance. Since $p_t$ is already known for each instant $t$, the LRT does not necessarily require measured values from previous instants. Therefore, $i_t$ and $d_t$ are adequate for the test’s input. The case of unknown packet-loss rate $\ell$ will be addressed later in the next section.

Since the binomial law belongs to the family of the exponential distribution, there exists a UMP test that is given by the following decision rule (see [4], Corollary 3.4.1):

$$\delta^*(d_t) = \begin{cases} \mathcal{H}_0, & \text{if } d_t \geq h, \\ \mathcal{H}_1, & \text{if } d_t < h, \end{cases}$$  

(4)

where $h$ is a threshold such that $\delta^* \in \mathcal{K}_{\alpha_0}$. However, we note that evaluating the statistical properties of this test, especially the false alarm and the power function, may be difficult. To simplify this task, it is proposed in this paper to apply the central limit theorem (CLT) ([4], theorem 11.2.5), assuming that the number of Interest sent $i_t$ is large, which is a very usual case for a router face. Hence, for legitimate traffic, the number of Data packets received can be modeled as:

$$d_t \overset{\text{I}}{\sim} N(i_t(1-p_t), i_t p_t (1-p_t)).$$  

(5)

On the opposite, when an IFA is started, the residual tends to:

$$r_t \overset{\text{I}}{\sim} N\left(a, \frac{p_t (1-p_t)}{i_t^2} N_a p_t (1-p_t) \right).$$  

(6)

For simplicity and clarity, it is proposed to denote $\sigma_r^2$ the variance under $\mathcal{H}_0$ and $\sigma_a^2$ the decrease of variance due to the IFA:

$$\sigma_r^2 = \frac{p_t (1-p_t)}{i_t}, \quad \sigma_a^2 = N_a p_t (1-p_t) = \frac{ap_t}{i_t}.$$  

(8)

We can note that the decrease of variance is due to the increase in number of Interest packets sent during the attack, i.e. from $i_t = i_t^*$ in Equation (8) to $i_t = i_t^* + N_a$ in Equation (9) while the number of receive Data packet does not change. From Equations (8) - (9), the testing problem (3) can reformulated as:

$$r_t \sim \begin{cases} N\left(0, \sigma_r^2 \right) & \text{under } \mathcal{H}_0, \\ N\left(a, \sigma_r^2 - \sigma_a^2 \right) & \text{under } \mathcal{H}_1. \end{cases}$$  

(9)

Equation (11) shows that the decrease of the packet-loss rate’s expectation $a$ characterizes the distribution of the residual under $\mathcal{H}_1$, hence it is used in the remaining of this paper to quantify the payload of the attack.

One can note that the function $f: d_t \rightarrow R(d_t) = r_t$ is strictly decreasing. Hence, the test $\delta^*(d_t)$ is equivalent to the following test:
\[
\delta^*(r) = \begin{cases} 
H_0 & \text{if } r_i \leq \tau^* = R(h), \\
H_1 & \text{if } r_i > \tau^* = R(h).
\end{cases}
\]  

As previously discussed, the application of the CLT \((8)\) allows establishing the statistical properties of the optimal UMP test, presented in the following Proposition 1:

**Proposition 1:** Assuming that the number of Interest \(i_t\) tends to infinity, for any prescribed false-alarm probability \(a_0\), the decision threshold \(\tau^*\), given by:

\[
\tau^*(a_0) = \Phi^{-1}(1 - a_0)\sigma_t,
\]

guarantees that the test \(\delta^*\) \((12)\) is in \(\mathcal{H}_{a_0}\). Here \(\Phi\) and \(\Phi^{-1}\) are the standard normal cumulative distribution function and its inverse function. Using the decision threshold given in \((13)\) the power function of the UMP test \(\delta^*\) \((12)\) is given by:

\[
\beta_{\delta^*}(a) = 1 - \Phi\left( \frac{\sigma_t\Phi^{-1}(1-a_0)-a}{\sqrt{\sigma_0^2-\sigma_a^2}} \right).
\]

### 4.1.5 Detection with unknown packet-loss rate

#### 4.1.5.1 Packet-loss Rate Model

This section addresses the case where the expected packet-loss rate \(p_e\) is unknown. In order to enhance the detection, \(N\) last measurements of packet-loss rate \(\ell = (\ell_{T-N+1}, \ldots, \ell_T)\) will be taken into account. Since the fluctuation of the packet-loss rate is limited and smooth \([5][6]\), its expectation can be modeled by a polynomial: \(p = Hx\), where \(H\) is a matrix of size \(N \times q\) with \(H_{ij} = t^{j-1}, i \in \{T - N + 1, \ldots, T\}, j \in \{0, \ldots, q - 1\}\) and \(x = (x_0, \ldots, x_q)\) is the vector of the \(q\) coefficients of the polynomial. Such a model has been widely used in signal processing, see \([7][8][9]\) for applications in Internet traffic modeling and image processing.

Assuming that packet-loss rate measurements \(\ell\) are independent, the CLT allows modeling those observations as follows:

\[
\ell \sim N \left( Hx, \Sigma_0 \right),
\]

where \(\Sigma_0\) is a diagonal covariance matrix whose element is \(\frac{p_t(1-p_t)}{t^2}\), \(t \in \{T - N + 1, \ldots, T\}\).

When an attack is started at instant 'Erreur ! Source du renvoi introuvable.' \(T^{th}\), the last samples \(d_t\) and \(i_t\) will be affected by the increase in number of Interest packets sent \((1)(1)\) and packet-loss rate \(\ell_T\) \((2)\). Hence under hypothesis \(\mathcal{H}_1\), as \(i_T\) tends to infinity, the packet-loss rate will tend to the following model:

\[
\ell \sim N \left( Hx - av_a, \Sigma_0 - \Sigma_a \right),
\]

where \(\Sigma_a\), as in Equation \((8)\), represents the decrease of variance due to the IFA that only affects the corrupted samples, and \(v_a\) represents the change in loss-packet rate due to the attack, for instance, \(v_a = (0,0,\ldots,0,1)^T\) when only the very last sample is corrupted.

In this paper, it is proposed to use the least square method to estimate the expectation of the packet-loss rate \(p\):

\[
\tilde{p} = H(H^TH)^{-1}H^T\ell,
\]

and consequently the estimated residuals \(r\) are defined, as in Equation \((8)\), as:

\[
r = \ell - \tilde{p} = H^r\ell,
\]

where \(H^r = I_N - H(H^TH)^{-1}H^T\), with \(I_N\) the identity matrix of size \(N\), represents the projection onto the orthogonal complement of the subspace spanned by the columns of \(H\).
4.1.5.2 Proposed Test for unknown packet loss rate

From the model of the packet-loss rate under each hypothesis (13) - (14), one can note that the testing problem with unknown packet-loss rate can be formulated as a choice between the following hypotheses:

\[
\begin{align*}
H_0: & \quad \tilde{r} \sim \mathcal{N}(0, \Sigma_0 H^T), \\
H_1: & \quad \tilde{r} \sim \mathcal{N}(a \tilde{v}_a H^+ \Sigma_0 H^T - H^+ \Sigma_a H^T),
\end{align*}
\]

with \(\tilde{v}_a = H^+ v_a\) the footprint of the IFA after estimating and then removing the expected loss-packet rate (15). Here it can be noted that, as previously discussed in Section 4.1.4, the IFA impacts both the expectation and the covariance of the residuals.

Obviously, designing an optimal test for the hypothesis testing problem (16) is challenging. In this paper, it is proposed to apply the UMP test designed in the case of a known packet-loss rate by replacing the residuals by the estimated ones, from Equation (15). This leads to the Generalized Likelihood Ratio that is defined by:

\[
\delta_\tilde{r}(\tilde{r}) = \begin{cases} 
H_0 & \text{if } \tilde{v}_a^T \tilde{r} \leq \tilde{r}, \\
H_1 & \text{if } \tilde{v}_a^T \tilde{r} > \tilde{r}.
\end{cases}
\]

The very interesting results this paper proposes is that, because it is possible to establish the statistical distribution of the Generalized Likelihood Ratio \(\delta_\tilde{r}(\tilde{r})\), one can establish analytically the properties of the proposed test.

From the distribution of the residuals \(\tilde{r}\), Equation (16), it is straightforward that:

\[
\tilde{v}_a^T \tilde{r} \sim \begin{cases} 
\mathcal{N}(0, s_0^2) & \text{under } H_0, \\
\mathcal{N}(a\|v_a\|^2, s_0^2 - s_a^2) & \text{under } H_1,
\end{cases}
\]

where the variance of the GLR under \(H_0\) is given by:

\[s_0^2 = v_a^T H^+ \Sigma_0 H^T v_a,\]

and, similarly, the decrease of variance of the GLR under \(H_1\) is defined by:

\[s_a^2 = v_a^T H^+ \Sigma_a H^T v_a.\]

Similarly to Proposition 1, we can establish the decision threshold and the power function of the proposed GLRT as following:

**Proposition 2:** Assuming that the number of Interest \(i_t\) tends to infinity, for any prescribed false-alarm probability \(\alpha_0\), the decision threshold \(\tilde{r}\) given by:

\[\tilde{r} = \Phi^{-1}(1 - \alpha_0) s_0,\]

guarantees that the test \(\delta_\tilde{r}\) (17) is in \(\mathcal{K}_{\alpha_0}\). Using the decision threshold given in (19), the power function of the UMP test (17) is given by:

\[
\beta_{\delta_\tilde{r}}(\alpha) = 1 - \Phi \left( \frac{s_a \Phi^{-1}(1 - \alpha_0) - a \|v_a\|^2}{\sqrt{s_0^2 - s_a^2}} \right). \tag{20}
\]

From the power function (20), one can note that the loss of optimality of the proposed GLRT is mainly caused by the factor \(\|v_a\|^2\). This is explained by the fact that a non-negligible proportion of the packet-loss rate’s change due to IFA will be modeled as part of the regular change due to legitimate traffic.
4.1.6 Numerical Results

4.1.6.1 Scenario setup

To verify the proposed detection, two sets of numerical results are presented. First, results obtained on data simulated under *Matlab* are presented to verify the sharpness of the theoretical findings. Then, *ndnSIM* - an open source NDN simulator, provided by the NDN project - is used to generate more realistic data. Indeed, *ndnSIM* faithfully implements the components of a NDN network which allows us to consider every aspect of the network [10]. Besides, in order to compare the performance of our approach to the existing ones, we reuse one of the topologies from [2] - a binary tree with 8 hosts, intermediate routers and one content provider for our evaluation. The experimental settings are also referred to our prior work [11] that uses the same topology.

In all of our simulations, the actual number of Interests sent is generated from a Poisson distribution whose mean value is drawn from a uniform random variable. In addition, the actual packet-loss rate follows an auto-regressive (AR) model. Such model has been extensively used to model both users’ requests evolution and packet-loss rate in computer network [6], [12] and can be easily implemented in *ndnSIM*. More precisely, the packet-loss rate is initialized at \( p_0 = 0.05 \), then following expectations of packet-loss rates are given by \( p_t = p_{t-1} + u \) with \( u \) the realization of a uniformly distributed random variable. To avoid computational problem, the sign of \( u \) is flipped if \( p_t < 0 \). Several values for those parameters have been tested and the obtained results show similar trends.

Finally, note that for the proposed GLRT, a set of 50 samples was used and the degree of the polynomial is \( q - 1 = 4 \), hence the matrix \( H \) has the size \( 50 \times 5 \). In all the figures, except Figure 13, it is considered that the quickest detection is desired, hence it is aimed at detecting if only the last sample is corrupted. In such a case the footprint of IFA on the packet-loss rate is characterized by \( v_a \) that has non zeros only on its last element, which give a footprint after packet-loss rate estimation \( \tilde{v}_a \) with \( \| \tilde{v}_a \|_2^2 \approx 0.6 \).

4.1.6.2 Numerical results on simulated data

Since one of the main goal of the proposed detection is to establish the statistical properties of the proposed GLRT, Figure 10 shows a comparison between the theoretical probabilities of false-alarm and detection power, given in Proposition 2, and the empirical ones. Even for threshold that corresponds to probabilities as small as \( 10^{-3} \), the empirical results match well the theoretically established ones. This observation is important as this guarantees a prescribed false-alarm probability in a practical situation. This also shows the sharpness of the theoretical findings and relevance of the proposed model.

Then Figure 11 compares the theoretical and empirical power as a function of the IFA payload \( \alpha \) for both optimal LRT and proposed GLRT. We also note that the power is computed with two prescribed false-alarm rates \( \alpha_0 = 0.01 \) and \( \alpha_0 = 0.1 \). Figure 11 again shows the relevance of the theoretical findings since empirical power functions match the theoretical ones. However, for low false-alarm probability as \( \alpha_0 = 0.01 \), the number of required samples is very large, hence, empirical results are slightly less accurate.
4.1.6.3 Numerical results on ndnSIM data

It is hardly possible to obtain real data from NDN routers, as NDN is not yet deployed at large scale. Hence, it is proposed to verify the relevance of the proposed approach on data as close to reality as possible by using ndnSIM. Figure 12 presents, similarly to Figure 10, the comparison between the proposed GLRT theoretical and empirical false-alarm probability as a function of decision threshold $\tilde{\tau}$. Because the actual values of packet-loss rates are unknown, the optimal LRT cannot be included in this comparison. We note from Figure 10 that the number of samples is much smaller since running ndnSIM is time consuming. However, even with this limited number of samples, the empirical probability of false alarm matches the theoretical one. This results is very important as its shows that the proposed approach remains accurate with data "as close as possible" from real ones.
Deliverable D2.2: Security monitoring of NDN through virtualized components

As discussed in Section 4.1.5.1, previous figures focus on the case in which only one sample is corrupted by the IFA. Hence, Figure 13 shows a comparison between the theoretical and the empirical power of the proposed GLRT for three number of corrupted sample, denoted $M$, 1, 3 and 7. As one would expect, the power increases with the number of corrupted samples. This result emphasizes that the proposed method can be adapted to focus on the quickest detection, hence aiming at detecting only if the last sample is corrupted at a cost of lower detection accuracy. On the other hand, it is also possible to increase the detection delay, hence focusing on the detection of several lasts samples corrupted by the IFA, to ensure a higher detection accuracy.
4.2 Detection of the Content Poisoning Attack (CPA)

4.2.1 Principal Component Analysis result and detection metric selection

As a follow-up of the CPA evaluation result in the deliverable D2.1, we performed a Principal Component Analysis (PCA) on our overall data-set. The main goal of PCA is to convert a dataset of related variables into combinations of linearly unrelated variables (i.e. principal components). Only the most important PC (i.e. ones that accounts for the most variance in the data) will be kept to represent the dataset.

Table 3 Values of the two firsts principal components with the label of associated metrics.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Core router</th>
<th>Access router</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td># additional Interests</td>
<td>% good hit</td>
<td>% bad hit</td>
<td>% miss hit</td>
</tr>
<tr>
<td>-0.1618</td>
<td>-0.0778</td>
<td>0.3913</td>
<td>-0.3891</td>
</tr>
<tr>
<td>0.4252</td>
<td>0.3178</td>
<td>-0.144</td>
<td>0.1085</td>
</tr>
</tbody>
</table>

The values of the first two components that account for 80.5% of the total variance of data are provided as rows of the Table 3. This table shows that the first component, accounting for 56.5% of measurements variance, is featured by a high impact on: (1) core router’s bad hits, resources waste; (2) access router’s bad hits, miss, resource waste; and (3) client’s metrics. As such, this first component represents the main expected impact of the CPA with the injection of bad Data in routers’ cache. Meanwhile, the second principal component, accounting for 24% of total data variance, shows its main impact on: (1) provider; (2) core router’s good hit and miss; (3) access router’s miss; and (4) client. This component exhibits the side effect of the CPA that prevents the routers from caching good Data, hence creating a higher rate of miss hit and traffic to the legitimate provider.

Figure 14 now presents the projection of individual measurements on these first two components (+ marks) as well as the mean projection for each scenario (solid arrows). The red, green and blue marks represent the measurement of bestroute, multicast, unsolicited scenario, respectively. The figure clearly shows that the components distinguish the unsolicited scenario from multicast and best route scenarios that exhibit the same operating mode. In the figure, the cyan circle points out the projections of the experiments with least attack impact (lowest attack rate). Similarly, the dashed arrow indicates the direction toward which the results move when the attack strength increases. The figure clearly shows that the unsolicited scenario has a specific footprint mainly captured by the first principal component. As expected, for this case the unsolicited Data creates a high rate of bad hit. It also shows that the bestroute and multicast scenarios have similar impact when the attack rate increases, mostly featured by the second principal component. Indeed, those scenarios create a higher rate of miss hit as the legitimate clients try to avoid the bad Data from caches while, on the contrary, the bad client tries to prevent the caching of good Data.
Deliverable D2.2: Security monitoring of NDN through virtualized components

Figure 14: Projections of the measurements on the two first principle components.

Despite its danger, we considered the unsolicited scenario as an implementation flaw, hence can be fixed easily with a patch of NFD in the future. Therefore from now on, we only address the cases of bestroute and multicast scenarios for CPA. Since the second principal component mainly characterises these two scenarios, it would be relevant to detect the CPA by using metrics related to this component, including provider’s number of additional Interests; core router’s miss and good hit; access router’s miss; and client’s metrics. Since clients are more likely to be compromised, hence can provide inaccurate metrics, its metrics should not be taken into account for the detection. Moreover, it is impractical for a router to differentiate the good and the bad Data (by verifying the signature) [13]. As a result, the metric good hit becomes irrelevant for the CPA detection. In short, we only consider the following metrics for CPA detection at the moment:

- Provider’s number of additional Interests;
- Core router’s miss count;
- Access router’s miss count.

4.2.2 Metrics’ distribution

Similarly to the IFA detector, the hypothesis testing theory will be leveraged to develop a detector against CPA. To build the test, one must first acquire the knowledge of the concerned metrics distribution. In this section, the result from a particular case will be shown, as a representative. Other experimented cases yield similar results.

We first simply plot the distribution of miss counts at the access router. The solid line in Figure 15 exhibits the Probability Density Function (PDF) of miss count at the access router. One can observe that the shape of the empirical result is similar to a “bell curve”, hence the normal distribution probably fits it well. Indeed, as shown in Figure 15, the plots of the estimated normal distribution are close to the empirical one, proving for our choice. This would give us an advantage in building the detector since the normal distribution has been well-studied and easy to estimate parameters. In addition, one can remark that the distribution under attack (red line) has greater mean and variance than the one without attack.

Figure 16 and Figure 17 show the similar fitness of metric’s distributions at core router and provider to the normal distribution. However, the fitness at provider is less precise than at router. Moreover, one can observe that the mean of Figure 15 is higher than one of Figure 16, indicating that there is more miss count at access router than at core router. This can be explained by the fact

---

1 Bestroute scenario, 5 poisoned contents over 10000 contents in the popularity, good client rate = bad client rate = 100 Interests /s
that access router is where the competition between good and bad clients occurs to get theirs preferred Data. On the other hand, by comparing distributions on different routers, one can deduce information about the location of routers along the path of CPA, whether close to the clients or close to the providers.

Another remark is that the overlap zone of $H_0$ (no CPA) and $H_1$ (with CPA) gets smaller when we move from provider to core router then to access router (Figure 15, Figure 16, Figure 17), implying that it gets easier to differentiate the two cases when we move towards access router. In another word, the detector based on hypothesis testing will achieve the best detection power at access router and lose its power gradually at core router and provider.

![Figure 15: Miss count’s distribution at access router](image1)

![Figure 16: Miss count’s distribution at core router](image2)

![Figure 17: Distribution of provider metric (number of Interests received)](image3)

### 4.2.3 CPA local detector

In our problem, the null hypothesis $H_0$ would imply that “There is no CPA at this router/provider” while the alternative hypothesis $H_1$ implies that “This router/provider is suffering from CPA”. Based on the conclusion in section 4.2.2, these two hypotheses can be rewritten formally as follows:

$$\begin{align*}
H_0 : x &\sim N (\mu_0, \sigma_0^2), \\
H_1 : x &\sim N (\mu_1, \sigma_1^2) \quad \text{if } \mu_1 > \mu_0, \sigma_1^2 > \sigma_0^2.
\end{align*}$$ (21)

where $x$ is the detector input at the location (miss counts for router and number of Interests received for provider). The constraint of parameters in $H_1$ being greater than ones in $H_0$ is actually what is observed in section 4.2.2. In addition, it is assumed that, for a specific location, the parameter $\mu_0$ and $\sigma_0^2$ are stable and can simply be estimated from the mean and the variance, respectively, of a dataset collected a priori at the corresponding router/provider. Hence, thanks to the Neyman-Pearson lemma [4], one can establish the UMP test as follows:

$$
\delta^*(x) = \begin{cases} 
H_0 & \text{if } \sum_{i=1}^{n} x_i \leq \tau, \\
H_1 & \text{if } \sum_{i=1}^{n} x_i > \tau.
\end{cases}
$$ (22)

where $n$ is the number of samples considered, and $\tau$ is the threshold. The statistical properties of this UMP test is presented in

**Proposition 3:** For any prescribed false-alarm probability $\alpha_0$, the decision threshold $\tau$ given by:
Deliverable D2.2: Security monitoring of NDN through virtualized components

\[
\tau = \Phi^{-1} \left[ \alpha_0 + \Phi \left( \frac{-n\mu_0}{n\sigma_0^2} \right) \right] \sqrt{n\sigma_0^2} + n\mu_0, \tag{23}
\]
guarantees that the test \( \delta^*(x) \) (22) is in \( \mathcal{H}_{\alpha_0} \). Using the decision threshold given in (23), the power function of the UMP test (22) is given by:

\[
\beta_\delta(\mu, \sigma^2) = \Phi \left[ \Phi^{-1} \left[ \alpha + \Phi \left( \frac{-n\mu_0}{n\sigma_0^2} \right) \right] \frac{\sigma_0}{\sigma_1} + \frac{(\mu_0 - \mu_1)\sqrt{n}}{\sigma_1} \right] \tag{24}
\]

According to (23), the value of threshold \( \tau \) only depends on \( n, \mu_0, \sigma_0^2 \) which all are parameters that can be chosen or estimated a prior. In another word, the threshold does not depend on the attack payload of CPA. Figure 18 illustrates the probability of false alarm as a function of the threshold at different locations. In general, the empirical result and theoretical one are pretty close, proving the relevance of the approach and the ability to guarantee the prescribed PFA \( \alpha_0 \) of our detector. Nevertheless, the result is not so good for the provider, especially with a small \( \alpha_0 \). This can be explained by the poor fitness to the normal distribution of provider’s metric that we presented in the previous section.

![Figure 18](image)

(a) Access router  (b) Core router  (c) Provider

Figure 18: Probability of false alarm as function of threshold at different locations

Figure 19 exhibits the ROC curves, i.e. detection power as a function of PFA, at different locations and with different value of sample size \( n \). The figure shows how much detection power can be improved (i.e. ROC curves move toward upper left corner) by using larger sample size. Moreover, increasing the sample size also narrows the difference between the empirical and theoretical result, making the model more accuracy. An interesting remark is that when moving from access router toward the provider of the target content, the performance of the detector is weakened. The reason for this phenomenon is because the overlap zone of \( \mathcal{H}_0 \) and \( \mathcal{H}_1 \) distributions get larger when we move along this direction (Figure 15, Figure 16, Figure 17). In the similar way, one can also conclude that the performance of the detector rises when moving from the provider toward the location of competing between good and bad clients (i.e. access router). Additionally, alarms from local detectors could be combined to achieve a more precise overall detection.
Deliverable D2.2: Security monitoring of NDN through virtualized components

(a) Access router  (b) Core router  (c) Provider

Figure 19: ROC curves at different locations, with different sample sizes 1, 3, 5.
4.3 Detection of a mixed attack scenario (NFV/NDN)

Virtualized OSs are almost identical to the real systems. This allows exploits designed for one to be applied to the other. In the case of VMs based on an OS containing vulnerabilities, this could allow an attacker to obtain privileged control of the VM and carrying out other exploits. The same holds for para-virtualization techniques that depend on the host’s OS that can also contain vulnerabilities. In the same way, vulnerabilities in the VIM can allow an attacker to break out of the isolation. And, this can lead to DoS attacks, VM or hypervisor system crashes, and illegal access to memory of guest VMs, or the host, making it possible to read, write or execute its content. Vulnerabilities on popular VIMs like Xen, VMWare and Linux KVM have been found and will continue to be found in every new release due to the complexity and amount of code involved.

All these attacks, if effective, allow accessing the VMs and, in turn, allow attacking basic NDN functions, e.g., modifying or corrupting routing tables (FIB and PIT), caches, requests, or even gaining information from the system, known as side-channel attacks, which can be exploited to break any protection used by the system including encryption. These attacks have been described in detail in the deliverable D2.1.

The virtualized infrastructure implemented in DOCTOR (relying on Docker containers) presents several attack possibilities. We took the example of an attack scenario in which a malicious NDN client manages to trigger a vulnerability on the container’s technology of a router VNF to take control of this NDN router. This can be used to inject fake NDN packets in the network. Then, once in the Docker container the attacker manages to escape its isolation and access other containers on the same host, in order to execute code in another container (for instance a NDN router further in the virtual topology). This can be used to progress further in the network, bypassing potential detection mechanisms in the NDN topology.

The multi-step attack, as illustrated in Figure 20, could happen as follows:
- 1: The attacker sends an HTTP request containing a payload to exploit a NFD vulnerability to the server. The payload is converted in a NDN Interest by the IGW and sent to First NDN node R1.
- 2: R1 routes the interest to a second NDN node R2, which is vulnerable. The vulnerability is exploited and attacker takes control of R2.
- 3: The attacker exploits a vulnerability in isolation to break out of R2’s container and take control of Host3.
- 4: The attacker uses its privileges on Host3 to execute code on R6.
- 5: A malicious payload is generated from R6 to exploit a remote vulnerability in the EGW. This allows the attacker to bypass R4 and R5 without compromising them.
- 3b: from R2, the attacker can generate fake NDN content, sent back to the IGW
- 4b: the IGW converts fake content to HTTP responses, sent to legitimate clients
Deliverable D2.2: Security monitoring of NDN through virtualized components

Such an attack scenario can be anticipated by the CyberCAPTOR attack graph engine based on the new rules defined in section 2: if it has access to the topology for both NDN and IP domains, to the vulnerabilities and their consequences, it can predict all possible attack paths. Then, actual happening of the attack can be monitored by MMT.

Indeed, MMT is running locally in each VNF, for instance in the different NDN routers and gateways, to analyze the whole network traffic. This allows detecting the network part of the attack (e.g., sending malformed NDN packets). Besides, by analyzing system logs (SYSLOGs) inside containers, we intend to monitor the different system calls that allow detecting the third step of the attack where the isolation is broken. This part is still in progress and will be integrated in dedicated tool called MMT-INFRA.
4.4 Detection of information leakage in NDN

Information leakage is one of the main security threats for companies in the Internet and it is mostly the result of Targeted Attacks as reported in several security reports [14]. In this section, we consider the case where an enterprise network is based on NDN architecture, connected to NDN Future Internet, and prepares a firewall between the enterprise and outside network.

4.4.1 Problem of Information Leakage in NDN

4.4.1.1 Information Leakage through Data Packets in NDN

The basic framework of our firewall assumes the following assumptions: A1) operators have the naming policy and routing policy for all the information assets inside the enterprise network, A2) operators can announce to the outside network what content name in the enterprise network can be accessed from the outside network, but do not have any right for the naming policies outside their network, A3) a malware has the list of clients of this company.

It is easy to prevent the information leakages through Data packets in NDN as follows: i) define the name prefix for public contents as “/company/pub/” (A1), ii) announce “/company/pub/” to the outside network (A2). In this case, if the malware names the leaked list as “/company/pub/ABC”, then the Interest packet can reach the malware from the outside network via the gateway. This can be blocked by the next rule. iii) Any Interest packet from the outside network shall not be relayed toward the inside network (A1).

Even though the malware sends a Data packet toward the gateway, the gateway cannot send it toward the outside network since there are no PIT entries for that Data packet. If the malware announces the leaked list as “/company/pub/ABC” inside the network, the gateway may cache this content during the communications among the nodes inside the network. This threat can be blocked by the next rule. iv) The gateway shall not relay an Interest/Data packet whose name prefix is “/company/pub/” from the inside network back to the inside network (A2). To realize the communications inside the network, the nodes utilize the name prefix such as “/company/priv/” for the private content, which cannot be accessed from outside.

In this case, all the publicly-accessible contents are on the gateway. To control the content of its own, the following rule is required. v) All the publicly-accessible content must be listed in the white list (A1). These five rules can shut out the information leakages through Data packets.

4.4.1.2 Information Leakage through Interest Packets in NDN

However, if a computer is compromised by an attacker's malware, it is therefore possible for the malware to use this computer to encode confidential information into the names of Interest packets (i.e., steganography embedded) and send these anomalous Interest packets out of the network toward the attacker.

There are two possible methods to perform information leakage through Interest packets, referred as “TYPE I” (1-way Interest packets) and “TYPE II” (Interest packets with their corresponding Data packets). These methods are summarized on Table 4 and illustrated on Figure 21 and Figure 22.

| Table 4 : Taxonomy of information leakage through Interest packets |
|-------------------------|-----------------|-----------------|
| Features                | TYPE I one-way Interest | TYPE II Interest and Data |
| Malware remote control  | No              | Yes             |
| Retransmission          | No              | Yes             |
| Attacker anonymity      | Yes             | No*             |
| Erasure coding          | Yes             | No              |
| PIT overflow            | Yes             | No              |

*Yes, for some cases (bots, etc.)
For TYPE I method (Figure 21), a malware transmits leaked information out of the enterprise network by sending Interest packets to the attacker. The malware has to know the name prefixes toward compromised NDN nodes (e.g., Wi-Fi AP installed by the attacker) in order to forward Interests to the attacker. In this method, the attacker does not reply with any Data packets to the malware and cannot have a fine-tuned control of the malware. Therefore, in the case that some Interest packets have been lost, the attacker cannot request for a retransmission of missing information. This can happen if a firewall drops packets, or when a PIT is overflowed. The attacker may use erasure coding such as LT codes [15] and Raptor codes [16] to deal with dropped Interest packets. Although this method is not the most efficient for leaked information, the main advantage of the method is to preserve the attacker anonymity because only anomalous Interests packets are received and the attacker does not reply with any Data packets with its own signature.

For TYPE II method (Figure 22), an attacker controls explicitly the malware and can communicate with it by using Interest and Data packets. Thus, the attacker can remotely control the malware assuming the name prefix of the attacker is routable from the malware. In the case of a packet drop, the attacker can request a retransmission with Data packets and there is no need to use erasure coding. This method is more efficient than the TYPE I method, but once the attack is detected, it is possible for the attacker to be tracked as his signature was included into Data messages. The attacker, however, can control bots remotely and avoid being tracked.

Note that in this report we consider only Name element of the Interest packet format for information leakage and we keep for future work other elements such as Selectors or Nonce [17].

![Figure 21: TYPE I: One-way Interest](image1)

![Figure 22: TYPE II: Interest/Data](image2)

![Figure 23: (a) URL [18]; (b) content name naturally extending URL](image3)
Mason et al. [19] proposed English Shellcode to encrypt and hide information, which transforms the Shellcode into the one similar to English prose. We show a similar design of encoder/decoder to include some information in names of Interest packets, assuming: i) the names in NDN will follow similar naming scheme as those of URLs in the current Internet and described by the RFC 1808 (RFC1808) (Figure 13 (a)); ii) the Interest name prefix of the attacker including <net_loc> part is routable from the malware. In this report, we set <net_loc> part and the name of application as “attacker.(TLD)” and “/info-leak”, respectively. For example, the name from <scheme> part to “/info-leak” should be “ndn://attacker.com/info-leak”, assuming an attacker creates anomalous names which belong to “com” domain.

![Diagram of Transmission of leaked information from malware to attacker](image)

Figure 24: Transmission of leaked information from malware to attacker

Figure 24 shows a general framework to transmit leaked data from a malware to the attacker. First, the data to be leaked is compressed (Zip). For TYPE I method, the malware further encodes the compressed data with erasure encoding. To bypass a firewall and realize information leakage with names, the malware further encodes the output data by our dictionary encoding for steganography. Then, it creates anomalous names to leak data. To perform dictionary encoding, we prepare a table with each dictionary word and its corresponding digits. The dictionary words used in this report will be introduced in the performance evaluation part (see Section 4.4.3).

### 4.4.2 Detection with Name-Based Filters

This report focuses on information leakage from compromised computers in an enterprise network. We therefore assume a regular user behavior to access outside content: i) users can access remote content, whose names are found by search engines, or by accessing links toward the content; ii) content access policy is managed by network administrators and they can prohibit accessing unwanted content names. Network administrators can also explicitly define White List with names that can be accessed.

#### 4.4.2.1 Name-Based Filter Using Search Engine Information

In the attack by steganography embedded Interest packets, anomalous names are created in order to leak information from corporate network (e.g., /mydomain/leak/name1/bank1/name2/bank2), and are not common names to be requested by users with search engines (e.g., /named-data.net/doc/ndn-tlv/).

Assuming that search engine services will still exist in future NDN network, search engines could serve to help detecting legitimate names. Thus, we propose first a name-based filter using search engine information. For instance, when our filter receives an Interest name, it performs a request to a search engine. If the name is indexed by the search engine, the filter considers the Interest packet...
Deliverable D2.2: Security monitoring of NDN through virtualized components

as legitimate; otherwise the filter drops the packet. This first approach is easy to implement but can also be considered as being naive regarding today's Internet use.

Indeed, according to [20], search engines index only 4% of all content (referred as Surface Web), and the remaining 96% are not indexed by search engine (referred as Deep Web). In addition, the number of Internet users using search engine decreases, as its amount was 55% in 2014, to only 49% in 2015 [21]. One of the reasons for this decrease is the mobile era, and the fact that it is difficult for users to search content with small-size mobile screen, and they prefer accessing content via other methods such as social networks. From company perspectives, it is usually prohibited for company users, who work with company-supplied personal computers, to use social networks by their working regulations in terms of labor productivity as well as information leakage prevention. Following the conventions described in this Section, the enterprise network administrator will have to define policy to allow users access remote content; search engine or authorized links toward content.

As most of content come from Deep Web (96% as stated before), many of them should still be accessible by company users. Thus, a huge amount of content is not indexed by search engines such as newly generated ones or password-protected ones. A simple name-based filter using search engine information will therefore not be accurate enough to decide the legitimate names to be accessed. Indeed, this filter cannot be aware of most content names that are not indexed. Thus, there is need to add further information to improve the efficiency of name-based filter. To overcome this proposal, we go one step beyond and propose a more sophisticated name-based filter using one-class SVM.

4.4.2.2 Name-Based Filter Using One-Class SVM

In order to filter out anomalous Interest packers used to leak information from a corporate network, we propose a name-based filter using one-class SVM [22]. One-class SVM is a very useful model to perform an anomaly detection in the case that there are not enough anomalous samples. It relies on unsupervised learning techniques that are commonly used with data mining.

Regarding NDN architecture, as it is not deployed, there are currently not anomalous traffic nor content names available. Thus, we rely on URLs that are commonly used on the Internet.

We thus consider NDN names as being URLs and we study the URLs properties in the next Section. Based on these characteristics, we will be able to describe feature vector and parameters for our name-based filter using one-class SVM. Attack scenarios (steganography embedded) with generated anomalous names will therefore be considered, and our countermeasures to detect if names are legitimate or anomalous will be evaluated.

4.4.3 Performance Evaluation

In this section, we evaluate the performances of the steganography embedded Interest packets attacks and show the throughput that can be achieved to bypass filters and leak information from the network.

4.4.3.1 URLs Dataset

NDN is an architecture for Future Internet and it has not been deployed at large-scale. This is not an operated network and there is no data set that is representative of regular use of this network. Thus, we consider names in NDN network will be based on URLs as stated before. This section presents therefore our URLs data set and describes in details the characteristics of the data set.

In order to infer the properties of names commonly used in the Internet, we collected URLs from the data repository provided by Common Crawl [23]. At first, we obtained the crawl archive for February 2016, which holds more than 1.73 billion URLs and we extracted unique URLs belonging to 7 Top-
Deliverable D2.2: Security monitoring of NDN through virtualized components

Level Domains (TLDs): “com”, “net”, “org” and “info” as gTLDs (generic Top-Level Domain), and “jp”, “fr”, and “uk” as ccTLDs (country code Top-Level Domain). In this data set, the number of URLs varies largely for each TLD (million to hundreds of millions). Thus, in order to obtain the same amount of URLs for each TLDs, we extract randomly 1 million URLs for each TLDs. Thus, in this paper, we rely on 7 million URLs and analyze their common characteristics in the following (1 million for each TLDs).

To configure our filters, we separated the 7 TLDs data set into two distinct sets: a training set and a testing set. Each training set contains 800,000 URLs, and it is used for the SVM to learn the classification rules. The remaining 200,000 URLs are used to test our filters.

4.4.3.2 Protector

4.4.3.2.1 Name-Based Filter Using One-Class SVM

We propose a name-based filter based on one-class SVM as presented in Section 4.4.2.2. We derive from the URLs properties 125 features for the one-class SVM method as follows: seven URLS attributes (Length of path, query, directory name, and file name, number of “/” in path, “=” and “&” in query), 26 alphabets characters in path and in query, and 33 other printable characters in path and in query.

For our SVM filter, we choose the “radial basis kernel” parameter because it is the most adapted to our data set [24]. With this kernel parameter, we made the models fit to the training set configuring the parameter $\nu$, which is the upper bound of the training errors, as 0.01, 0.05, 0.1, 0.2, 0.3 and 0.4, and also the parameter $\gamma$, which is kernel coefficient, as one divided by the number of dimensions in the feature vector (i.e., 1/125), which is the default value in scikit-learn [25]. On the one hand, if $\gamma$ is very large, it can lead to over-fitting. On the other hand, if $\gamma$ is very small, the model can be similar to linear model. It means that the training and testing errors depend on the value of $\gamma$.

Following above processes, as a result, the differences between training and testing errors were quite small, and the training errors were close to the $\nu$ parameter. As training and testing exhibited the same low errors, we could generalize the results and the false positive rate will be close to $\nu$ value. Note that false positive rate is defined as the ratio of the number of legitimate content names identified as anomalous divided by the total number of legitimate content names.

4.4.3.3 Attacker

4.4.3.3.1 Leaked Data

In order to leak the data, we prepared three Pdf files (Y.4001/F.748.2, Y.4412/F.747.8, Y.4413/F.748.5) from latest ITU-T recommendations [26]. These Pdf files include a variety of text and figures, which are common in technological documents. Specifically, in Y.4001/F.748.2, Y.4412/F.747.8 and Y.4413/F.748.5, there are 6, 4 and 9 figures and 18, 22 and 24 pages, respectively. Then, we compressed and converted these files into a single Zip file (3.4 MBytes). Hereafter, we consider how an attacker obtains this file. In our experiment, the Zip file is directly encoded into the names in Interest packets by dictionary encoding without erasure encoding.

4.4.3.3.2 Anomalous Name Creation by Attacker

Figure 25 shows the detailed flow to create anomalous names with the dictionary coding (i.e., steganography) in “com” TLD. As an attacker wants to add as many meaningful words as possible into each name to increase the throughput of the leaked information, the main principles to create anomalous names are (i) to choose the legitimate URL whose length is long enough; (ii) then to extract the name features (such as length of path); and (iii) to make the anomalous names similar to the selected URL. We assume that the attacker prepares a data set independently from the protec-
Deliverable D2.2: Security monitoring of NDN through virtualized components

tor and builds its one-class SVM filter by using this dataset and creates information-leakage packets with anomalous content names.

![Flowchart](image)

**Figure 25:** Flow to create anomalous names to leak data in the “com” domain (Explanation about \( \nu \) is shown in Section 4.4.3.2.1).

In order to prepare a table with each dictionary words and its corresponding digits (steganography), we extracted the words from WordNet [27]. WordNet 3.1 counts 147,478 words. In terms of the transmission rate, it is better to select shorter words. Therefore, we sort the 147,478 unique words in ascending order of their length, we extract the top \( 16^4 = 65,536 \) words, and we assign 4 hexadecimal digits to each word in our conversion table.

### 4.4.3.3 Per-Packet Throughput of Information-Leakage

![ROC Curve](image)

**Figure 26:** ROC curve for dictionary encoding with \( \nu=0.4 \)
 Deliverable D2.2: Security monitoring of NDN through virtualized components

Figure 27: Actual false positive rate

Figure 26 shows the ROC curves when the names by dictionary encoding designed for the thresholds at $\nu = 0.4$ are processed by the filters designed for $\nu = 0.01$ to 0.4. While the True Positive Rate (TPR), which is defined as the ratio of the number of anomalous content names identified as anomalous divided by the total number of anomalous content names, all false positive rates are the testing errors for the filters using one-class SVM. Fig. X shows that all the TPR are almost 0 for $\nu < 0.2$ except for the “fr” domain.

If our filter is not used, the attacker can fill the URL with the leaked data in hexadecimal digits. Selecting the longest URL in our data sets (4,127 characters excluding FQDN), the per-packet throughput of information-leakage reaches 2.06 KBytes/Interest_packet, which is maximum since 1 Byte of leaked data is mapped into 2 hexadecimal digits.

If our filter is used, as shown in Section 4.4.3.3.2, the attacker has to encode two Bytes of leaked data into each word used in the name template (e.g., ndn://attacker.com/info-leak/word1?key1=word2&key2=word3&key3=word4&key4=word5). Thus, as an example, our created name template conveys at most 10 Bytes of leaked data because all the words in the template may not be used due to the threshold restrictions such as length of path. When $\nu$ for the one-class SVM filter is set to less than 0.2, our SVM filter can start filtering out some of these packets. For each name template, regarding previous example in Section 4.4.3.3.1 with 3.4 MB of leaked data, one can compute the per-packet throughput by summing up all the received leaked data bytes and dividing this sum by all the sent packets. The average of per-packet throughput for all the domains except “fr” gives the information-leakage throughput in our data set, which is up to 7.79 Bytes/Interest_packet. We do not include “fr” domain as its TPR was high compared with other domains (Figure 26). Thus, by using our filter, the malware has to send 264 times more Interest packets to the attacker than without using the filter (2.06 KB/7.79 B).

Note that the false positive rates shown in Figure 27 are computed only for the filter using one-class SVM. Actual false positive rate should be computed after checking content names with the name-based filter using search engine information and one-class SVM. Thus, the actual false positive rate is $\nu \times R_{NotIndexed}$. $R_{NotIndexed}$ is the probability that legitimate users access real Deep Web content, which are not indexed by search engine. Figure 27 shows the actual false positive rate for each $\nu$ depends on $R_{NotIndexed}$, which will be very small if the enterprise network is properly managed. Therefore, we can keep low false positive rate and high performance for our filter.

Although it is possible for information to be leaked from NDN network through Interest packets, our proposed filter chokes off per-packet throughput of information-leakage. Therefore, the malware has to send a huge amount of Interest packets to the attacker to leak information and the throughput of this threat is drastically reduced. Thanks to our filter, it will be easy to detect anomalous Interest traffic by a per-flow filter based on traffic analysis at the subsequent stage. Network administrator can also perform naming policy control in NDN and reduces risks of information-leakage.
5 Conclusion

This second and last deliverable of the security-centric task of the project was more precisely about security monitoring, and had for objective to define the monitoring architecture and specific monitoring functions that will be used in order to secure the deployment of our virtualized NDN architecture. We described this architecture at three levels, from the more general view to the more precise one. First, we defined new rules in Mulval to make it able to consider new threats regarding the disruptive NFV and NDN network environments, so that we can later conduct a security analysis to identify the most critical attack paths that could affect our architecture and monitor it accordingly. Second, we described our monitoring architecture, more precisely what metrics we monitor and how the monitored data are collected and processed thanks to Montimage distributed probes and operator. Finally, we dig into the details of three precise detectors that can process the data we collect to detect NDN attacks and we showed their efficiency through extensive experiments.

All the developed detectors were published in the research community to improve the capacity of the future NDN monitoring tools to detect these critical attacks. Also, the two companies of this project consortium have included new rules in their respective security products to manage NDN security threats.

DOCTOR’s overall architecture (WP1) and monitoring architecture for security (WP2) being defined and implemented (WP4), the next step of the project consists in taking advantage of our virtualized architecture to design and develop an orchestrator (WP3) able to react in real time to the network to protect ensure the security and quality of service. The direct continuation up of this deliverable is to make the orchestrator trigger the proper actions as counter-measures when an attack is detected through automated MANO operations leveraging SDN and NFV capabilities. Also, in the case of more complex attacks, the attack graph analysis performed thanks to Mulval should be able to lead the orchestrator’s actions based on the probability of the different attack paths ongoing.
6 References


Deliverable D2.2: Security monitoring of NDN through virtualized components


