SEVENTH FRAMEWORK PROGRAMME

Information & Communication Technologies
ICT

Cooperation Programme

Nippon-European Cyberdefense-Oriented Multilayer threat Analysis
[†]

Deliverable D4.1: Requirements and specifications of testing environments

Abstract: This deliverable presents the testing environments, covering the essential requirements about their construction and the specifications of the demonstrators for each scenario, as well as the indicators specified for the respective experimental use cases.

Contractual Date of Delivery	February 28, 2015
Actual Date of Delivery	February 28, 2015
Deliverable Dissemination Level	Public
Editor	Gregory Blanc and Yuji Sekiya
Contributors	All *NECOMA* partners

The *NECOMA* consortium consists of:

Institut Mines-Telecom	Coordinator	France
ATOS SPAIN SA	Principal Contractor	Spain
FORTH-ICS	Principal Contractor	Greece
NASK	Principal Contractor	Poland
6CURE SAS	Principal Contractor	France
Nara Institute of Science and Technology	Coordinator	Japan
IIJ - Innovation Institute	Principal Contractor	Japan
National Institute of Informatics	Principal Contractor	Japan
Keio University	Principal Contractor	Japan
The University of Tokyo	Principal Contractor	Japan

[†]The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7-ICT-2013-EU-Japan) under grant agreement n° 608533.

Contents

List of Figures

1 Introduction

Past research projects focused on either data acquisition, analysis or cyberdefense, without enough consideration and connection to the upstream or downstream processes. These processes therefore existed in isolation, without much to be done on operational infrastructure. The aim of *NECOMA* project is to facilitate pipelining of data acquisition, analysis and cyberdefense by considering requirements of the downstream process and limitations of the upstream process. As gaps got identified in pipelined processes, our experts at different stages of the pipeline (from workpackage 1 to workpackage 3) have been working together to extend the capabilities of each process and create new capabilities as necessary. As a result, automated defense and mitigation against certain classes of attack patterns are made possible, whose capabilities will be benchmarked and demonstrated over research networks and/or testbeds.

Our ambition, in addition to proof-of-concept study, prototype development, and testbed experiments, is to validate the effectiveness and usability of our designs in practical use cases, ensuring that the desirable security properties of those information assets can be preserved. The worst case is that the performance of those services of interest can be maintained to an acceptable level, e.g., meeting the explicit requirements of corresponding SLAs, in the presence of attacks by improving the resilience of existing defensive mechanisms.

In general, the experimental use cases attempt to cover a broad scope of the threat landscape in line with commonly deployed policy enforcement points. So the initial step, done in workpackage 3, is to carry out a survey at partners site over the PEPs to determine what enforcement mechanisms will be deployed, and what type of input they need to properly respond to the attacks. In particular, four use cases will be constructed, including DDoS mitigation, botnet introspection, smartphone user protection and malware campaign mitigation.

Organization. This deliverable introduces each of the use cases in turn, with a brief description of the threats addressed and the proposed countermeasures, as well as a collection of scenarios that put into practice resilience mechanisms described in deliverables D3.2 and D3.4. Namely, Chapter 2 deals with *DDoS Mitigation*, while Chapter 3 covers *Botnet Introspection*. *Smartphone User Protection* and *Malware Campaign Mitigation* are addressed in Chapters 4 and 5, respectively. For each use case, we have collected a number of metrics that may be applicable to assess the success of the resilience mechanisms. Actually, several partners have contributed to explore the use cases with a specific scenario, that may cover a different scope, address a different threat or protect a different asset. Given these characteristics, additional scenario-specific metrics have been proposed to provide a more accurate evaluation. Finally, Chapter 6 describes specifications of the platforms that will host the several demonstrators based on the requirements and metrics introduced in the earlier chapters.

2 DDoS Mitigation

Distributed denial of service attacks can be mitigated in a number of ways, by either tackling the source of the attack, shaping the traffic, or degrading the service, among other solutions. However, none of these is a silver bullet. In this use case, we will recreate the situation of stress endured by information systems under attack, with an effort to experimentally verify the efficiency of proposed methods on identifying and blocking anomalous network flows.

2.1 Description

Distributed Denial of Service (DDoS) attacks have been extensively studied for more than two decades, but a surge in their growth can still be witnessed. In particular, flooding-based attacks, such as UDP, TCP SYN and ICMP floods dominate the growth, and the target of such volumetric attacks is to deplete computing resources like CPU, memory and network bandwidth by sending forged packets. Recent years have also seen an increase in multi-vector DDoS attacks [2]. One example is that of a large UDP flood combined with a slow HTTP GET flood [5], misleading victims to cope with the seemingly anomalous UDP flood, while the HTTP flood can slowly deplete the HTTP server computing resources.

To cope with DDoS attacks, tremendous efforts have been made from both academia and industry [11, 20]. However, few of the existing DDoS mitigation techniques have been considered for widespread deployment, primarily because of their implementation and deployment complexities, as well as prohibitive operational costs. One of the major reasons is that they usually require large network connection state tables to be maintained at routers or switches, resulting in extra storage and computational burdens. Also, some techniques like packet marking [14, 3] require a huge amount of packets to be monitored and collected, incurring additional processing over-

head. More importantly, the operation of those techniques relies on the deployment of additional modules or devices, increasing deployment complexity. Overall, such strong design and deployment assumptions indicate a lack of *autonomic* properties, causing non-trivial labor cost and response latency. Despite early efforts on designing autonomic DDoS response [17, 6], their scalability and operational costs are questionable in the face of large-scale deployment, mainly due to the intensive collaboration and communication between different detection modules that must be installed in advance.

In this use case, we focus on several scenarios that attempt to address the limitations discussed previously. In particular, our resilience mechanisms do not need manual reconfiguration thanks to technologies that are either in widespread use (such as MPLS) or emerging (such as SDN). The different scenarios addressed in this use case affect different kinds of assets (customer networks, internet exchange points), and tackle a diverse set of attacks (flooding, amplification). However, they all pursue the same goal which is to ultimately nullify the effects of DDoS attacks, or at least reduce their impact against the infrastructure or the end-user. The different scenarios attempt at representing the diversity found in real-life cases.

2.2 Metrics

In this section, we define evaluation metrics for DDoS mitigation mechanisms. We do not consider that these metrics are applicable to all scenarios, since these scenarios assume different mitigation mechanisms. Accordingly, we should select applicable metrics for a given scenario.

The metrics presented below are classified in four major categories. The first category deals with *Effectiveness*: in this category, metrics evaluate how accurately attacks are mitigated by the mechanism. The second category mentions metrics related to *Overhead*: the metrics are used to reveal how the mechanism performs from various perspectives. Last categories assess the *Cost* of deploying and maintaining the mechanism: the metrics are not derived from measurement results as it is the case for the above metrics, but still describe characteristics of the assessed mechanism.

Each metric is described below with its type, definition, score, and measurement methodology.

2.2.1 General Metrics

2.2.1.1 Effectiveness

2.2.1.1.1 Recall/Precision

Type: Quantitative

Definition: How correctly malicious packets are filtered or blocked by a mechanism.

Score: Recall (0 - 1), Precision (0 - 1), F-measure (0 - 1)

Measurement: sends sets of legitimate and malicious packets through the mechanism and captures raw packets at both ingress and egress points of the mechanism. After the packet capture, the two set of packets (inbound and outbound) are compared. Then counting how many legitimate and malicious packets are filtered on the mechanism.

2.2.1.1.2 Specificity

Type: Quantitative

Definition: How efficiently legitimate packets are preserved (effectively routed to the victim host, i.e., not blocked or filtered) by a mechanism. It may be seen as the legitimate packet loss rate.

Score: (0-1) (= [legitimate pps at the ingress] / [legitimate pps at the egress])

Measurement: measure legitimate traffic in terms of number of packets at the edges of the mechanism.

2.2.1.2 Overhead

2.2.1.2.1 Time to Recover

Type: Quantitative

Definition: How quickly a mechanism responds to an attack.

Score: t [second] ($t > 0$)

Measurement: measure the time from when the mechanism is initiated until the traffic is mitigated or dropped.

2.2.1.2.2 Latency

Type: Quantitative

Definition: Latency caused by the proposed mechanism.

Score: t [second] ($t > 0$) (= [latency with the mechanism] - [latency without the mechanism])

Measurement: measure latencies from legitimate host to victim host, with and without the mechanism.

2.2.1.2.3 Maximum Flow Capacity

Type: Quantitative

Definition: The maximum number of flows which can be handled by the mechanism.

Score: n [flows]

Measurement: Traffic generators are employed for this measurement. The generator makes multiple TCP sessions from the server to the client through a mitigation mechanism and increases the number of session and measure maximum sessions successfully established.

2.2.1.2.4 Simultaneous Flow Capacity

Type: Quantitative

Definition: The maximum amount of simultaneous flows and the rate of flow establishment handled by the mechanism.

Score: r [new flows per second]

Measurement: Traffic generators are employed for this measurement. The generator makes multiple TCP sessions from the server to the client through a mitigation mechanism and increases the number of session and measure how many sessions are successfully established per second.

2.2.1.2.5 Throughput

Type: Quantitative

Definition: How much traffic can be handled by the mechanism per time unit.

Score: x [bps] and x [pps]

Measurement: measuring throughput through the mechanism.

2.2.1.2.6 CPU Load

Type: Quantitative

Definition: CPU load of the mechanism when changing traffic or session volume.

Score: Traffic Volume vs CPU Load, Session Volume vs CPU Load

Measurement: measuring CPU loads of the mechanism with various traffic and session volume.

2.2.1.2.7 Memory Usage

Type: Quantitative

Definition: Memory usage of the mechanism when changing traffic and session volume with the mechanism.

Score: Traffic Volume vs Memory Usage, Session Volume vs Memory Usage

Measurement: measuring memory usage of the mechanism with various traffic and session volumes.

2.2.1.3 Cost

2.2.1.3.1 Installation Cost

Type: Qualitative

Definition: How easy or difficult it is to deploy the mechanism in the network in terms of topology changes or service stoppage.

2.2.1.3.2 Operation Cost

Type: Qualitative

Definition: The cost of operating the mechanism in the network and related overhead to daily operations.

2.2.1.3.3 Failure Cost

Type: Qualitative

Definition: The difficulty to recover from failures of the mechanism.

2.2.1.4 Deployment

2.2.1.4.1 Target Network

Type: Qualitative

Definition: To which type of networks can the mechanism adapt.

Category: ISP, Enterprise, Home, Datacenter,...

2.2.1.4.2 Locality

Type: Qualitative

Definition: How closely does the mechanism work to the attack source.

Category: Provider Edge, AS Border, Home Gateway, Enterprise Gateway,...

2.2.1.4.3 Stealthiness

Type: Qualitative

Definition: Detectability of the mechanism to attackers using active or passive measurement, and the awareness of such detection.

2.2.2 Scenario-specific Metrics

2.2.2.1 Time to start video

Type: Quantitative

Definition: How much time it takes for the initial buffering to be completed and start the video.

Score: t [time in seconds]

Measurement: measured by the video streaming client.

2.2.2.2 Time to rebuffer video

Type: Quantitative

Definition: How much time it takes to rebuffer the video when the streaming buffer gets empty due to low networking conditions.

Score: t [time in seconds]

Measurement: measured by the video streaming client.

2.2.2.3 Average video level (bitrate) downloaded by the video client

Type: Quantitative

Definition: How the quality of the video is adapted based on network conditions

Score: l [in bps]

Measurement: measured by the video streaming client.

Table 2.1: Metrics to be collected for DDoS mitigation scenarios

Common metrics		
Metric	Description	Units
Recall/Precision	accuracy of detection	%
Specificity	legitimate packet loss rate	%
Time to Recover	the time from detection of the attack to the effective mitigation	second
Latency	the time caused by the proposed mechanism for legitimate traffic	second
Maximum Flow Capacity	the maximum number of flows which can be handled by the mechanism	flows
Simultaneous Flow Capacity	the maximum amount of simultaneous flows and the rate of flow establishment handled by the mechanism	flows per sec
Throughput	how much traffic can be handled by the mechanism per time unit	bps/pps
CPU Load	CPU load of the mechanism when changing traffic or session volume	-
Memory Usage	memory usage of the mechanism when changing traffic and session volume with the mechanism	bytes
Installation Cost	how easy or difficult it is to deploy the mechanism in the network in terms of topology changes or service stoppage	-
Operation Cost	the cost of operating the mechanism in the network and related overhead to daily operations	-
Failure Cost	the difficulty to recover from failures of the mechanism	-
Target Network	to which type of networks can the mechanism adapt	-
Locality	how closely does the mechanism work to the attack source	-
Stealthiness	detectability of the mechanism to attackers using active or passive measurement, and the awareness of such detection	-
Starting time	Initial amount of video buffering time	second
Rebuffering time	Amount of paused time needed to fill the empty video buffer	second
Average bitrate	Average quality of streamed video	bps

2.3 Scenarios

2.3.1 Pushing defense upstream

2.3.1.1 Description

In this scenario, we place ourselves in a situation where the DDoS attack traffic saturates *resources* close to the target as the attack traffic converges towards the victim. The objective is to be able to *distribute the defenses upstream* from the choking point in order to gain *headroom* at target level.

By saturating resources, we mean, for example, network links and equipment, but also potential on-premise defense mechanisms, such as firewalls, DDoS mitigation systems, etc.

By distributing the defenses upstream we mean being able to filter out at least part of the attack – we consider that there is no need to be able to filter all the attack by the distributed mechanism, because we suppose that we have on-premise defense mechanisms at our disposal as well.

By headroom, we mean gaining sufficient effect so that on-premise mechanisms can function. Depending on the situation, this might mean just avoiding the link or router saturation (i.e., the bottleneck was the network) or avoiding the saturation of the on-premise defense mechanisms in case their capacity is lower than the network's.

The scenario studies the use of MPLS to achieve the upstream distribution of the defenses. The idea behind this is to:

- build on existing, standard technology in order to maintain compatibility with already deployed equipments;
- use the intrinsic functionality in MPLS to segregate legitimate traffic from suspicious traffic;
- enforce lower quality of service for suspicious traffic and thus ensure that suspicious traffic suffers more from forwarding decisions made by routers than the legitimate traffic when available resources on the traffic path become saturated.

The attacks most likely to cause resource saturation before the on-premise defenses are amplification attacks. There have been several devastating, highly mediatized examples in the recent past such as the 400 Gbps NTP amplification[1] and 300 Gbps DNS amplification[2] attacks on CloudFlare's clients.

[1] https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/

[2] https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/

By their amplifying nature they are suited to generating large volumes of traffic, measured in bits per second (bps), but at the same time likely to remain more modest in terms of packets per second (pps). Large bps values are obtained when a small request by the attacker results in a large response that will be sent to the victim typically by using spoofed source addresses. Even if the response might be so large that one frame carrying the request will result in several frames carrying the different IP fragments, the pps rates will remain relatively low – the large frame sizes saturate the link bandwidth with a relatively small pps rate. Such attack mechanisms rely on the server application to relay the attack and are likely to use limited (even if large) set of vulnerable servers, each of which will be used to generate a large amount of responses.

Amplificative and/or reflective attacks can be considered having two victims: the actual target but also the amplifiers/reflectors themselves whose resources (CPU, RAM, outbound bandwidth, etc.) can be exhausted as well. Our focus in this scenario is on the primary victim who receives the responses from the amplifiers.

In other words, the previous two paragraphs provide the rationale for the type of attacks we are defending against in this scenario:

1. Large frames
2. Potentially fragmented
3. UDP-based
4. From non-spoofed sources (amplifiers)
5. From a limited number (order of 10^1 to 10^4) of sources

2.3.1.2 Requirements of the Testing Environment

Next we list some high level requirements for different elements making up the scenario architecture.

2.3.1.2.1 Network This scenario requires a set of MPLS-capable nodes that relay both legitimate and DDoS attack traffic to the targeted service. The number of nodes should be sufficient to provide at least two alternative paths between the attacker/legitimate user and the service.

2.3.1.2.2 Service (victim) The node hosting the targeted service needs to be able to respond to ICMP echo requests in addition to application layer requests (e.g., an HTTP server), both being used to verify the reachability and latency.

2.3.1.2.3 Legitimate user This node should be physically separated from the attacker nodes to avoid producing measurement artefacts if the attack generation saturates the attacker nodes.

2.3.1.2.4 Attacker We need one or more nodes capable of reinjecting the attack traffic (as if it were coming from amplifiers, e.g., DNS server, NTP server). As the traffic reinjection requires a lot of resources on the injection nodes, they should be physically separated from the legitimate users to avoid perturbing the measurements for legitimate traffic.

2.3.1.2.5 Metrics The evaluation includes the use of the following metrics for this scenario:

Specificity is an important metric for this scenario as it allows us to evaluate the amount of legitimate traffic the mechanism preserves. Indeed the mechanism's main purpose is to ensure the avaibility of the targeted service. We can evaluate the specificity for two different aspects:

- The segregation capability between malicious and legitimate traffic, for which the false positives are defined as legitimate traffic being forwarded as malicious.
- The behavior of the mechanism as a whole, for which false positives are defined as legitimate traffic that is effectively dropped by the mechanism. As the mechanism does not necessarily drop traffic considered as malicious, but forwards it with a degraded Quality of Service (QoS), part of the legitimate (and malicious) traffic can be routed all the way to the target.

Both specificity types will be evaluated.

Time to Recover to measure the time from detection of the DDoS attack (e.g., obtention of suspicious source IP addresses potentially taking part in the attack) to the effective mitigation of the attack traffic by the MPLS routers. This will include the time required for establishing the suitable configuration, its deployment and the time for the active configuration to have an impact on malicious traffic.

Latency to measure the latency of legitimate traffic with and without activating the defense mechanism; under three different conditions: nominal traffic, under non-saturating attack, and under saturating attack. In the case of nominal traffic we would obtain the latency overhead caused by the mechanism as such; in the case of a saturating attack we expect to gain in latency for legitimate traffic even with the overhead; and in the case of a non-saturating attack to be somewhere in between.

Precision/Recall to measure the amount of suspicious packets treated as such by the mechanism (not necessarily dropped packets, but packets being forwarded with lower QoS by the MPLS routers). We do not necessarily require very high values, as we place ourselves in a situation where the MPLS-based mechanism is followed by an on-premise mitigation mechanism; and the main objective is to gain headroom for this mechanism to be able to function and clean up the remaining attack traffic.

2.3.1.3 Topology Diagram

The current vision of the scenario topology is provided in Figure 2.1. This is still subject to changes as the scenario, its implementation, and the WP3 work providing the defense mechanism, become mature.

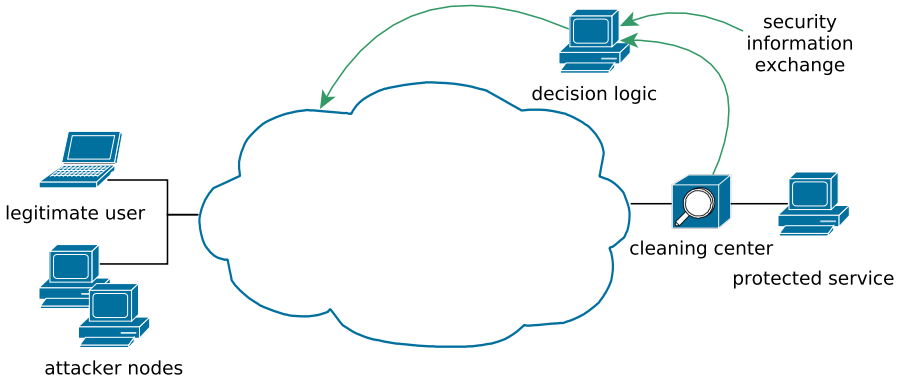

Figure 2.1: The current view of the scenario topology with the traffic generating nodes on the left, the targeted service and on-premise protections on the right, and the MPLS network in the center

2.3.2 SDN-based DDoS Mitigation

2.3.2.1 Description

Large scale DDoS attacks are a general issue on the Internet, even more so when the volume of the attacks is growing. Some of the attacks saturate not only the links at victim networks but also links at the Internet backbone such as transit networks and Internet eXchanges (IXs).

In this scenario, we demonstrate a simple SDN-based mitigation mechanism against Domain Name System (DNS) amplification attacks. The mechanism is comprised of an OpenFlow controller and a switch. Also, we use an sFlow monitoring system as a detector of the amplification attacks. When the monitoring system detects traffic of which volume is over a certain

threshold, the system posts the information to the NECOMAtter streams. The NECOMAtter is a knowledge sharing platform, spreading cyber threats information among bots (automated agents) and humans in a way similar to Twitter. When information about a threat is posted, a network operator who consumes the feed may initiate a mitigation process against the posted attack through the SDN controller. Accordingly, we can block the attack on deployed OpenFlow switches, and not only at the edge of the network.

2.3.2.2 Requirements of the Testing Environment

We describe the technical requirements for the scenario. We need some hosts and switches for the demonstration as follows.

2.3.2.2.1 Network This scenario requires a set of IP-capable networks. The networks should be comprised of three segments. One segment will host emulated attackers. The segments have to be connected with switches. Specifically, an OpenFlow-capable switch should bridge the attacker and the victim segments.

2.3.2.2.2 Attacker hosts The hosts generate malicious DNS queries, with the source IP address of the packets spoofed as the victim node. These nodes should be located in a different network segment from the victim node. Also, we require amplification nodes, which works as open resolvers to respond to the spoofed DNS queries from external networks.

2.3.2.2.3 Detector The node used to detect emulated DDoS attacks. The detector should be able to output information of attacks. When a DDoS attack is detected, the node has to share the attack information. The information contains source IP addresses of the attack.

2.3.2.2.4 Mitigator The node used to initiate a mitigation against an attack on the network. It should start with an attack information that is shared via the information sharing platform.

2.3.2.2.5 Victim host The node hosting the service targeted by the attacks in this scenario. The node hosts a web service that displays a simple web page to the users. Also, the server should be able to reply with ICMP packets in order to check for reachability.

2.3.2.2.6 Information sharing platform The platform used to share attack source addresses between the detector and the mitigator. At the very least, the platform should be able to share IP address of an attack source in text format.

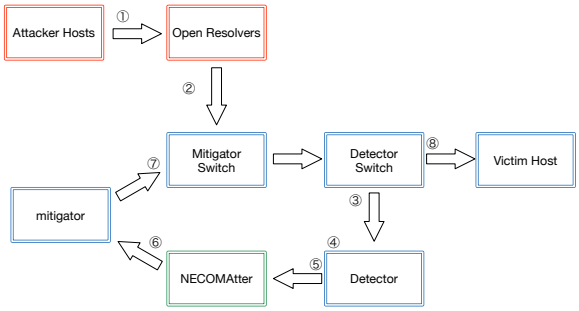

Figure 2.2: Scenario sequence for the SDN-based Mitigation against Amplification Attacks

2.3.2.2.7 Metrics We apply the following two metrics to evaluate the mitigation mechanism.

Time to recover to measure the time from the detection of DDoS attack to the effective mitigation of the attack traffic by the mitigator node. DDoS attacks have to be quickly filtered on the networks to minimize downtime of the victim services. The measurement results show how quickly the mechanism responds to attacks.

Latency to measure the latency of legitimate traffic with and without the SDN-based mitigation mechanism. The mitigation mechanism may intercept legitimate traffic as well as attack traffic. Accordingly, the mitigation itself will affect legitimate traffic. The measurement reveals additional latency for the legitimate traffic caused by the mechanism.

2.3.2.3 Sequence Diagram

Figure 2.2 shows the diagram of the scenario sequence. The flow of the scenario is as follows:

1. Attacker hosts send spoofed DNS queries to Open Resolvers.
2. Open Resolvers reply with DNS answers to the spoofed target.
3. Detector Switch sends flow information to the Detector.
4. Detector identifies malicious flows by utilizing some algorithm.
5. Detector posts the information of malicious flows on NECOMAtter, which is an information sharing platform developed in this project.
6. Mitigator monitors NECOMAtter timeline and picks up related information regarding amplification attacks, and decides to install mitigation rules into the Mitigator Switch.

7. Mitigator Switch mitigates only the attack (malicious) traffic, and allows legitimate traffic to pass through.

2.3.3 DDoS Mitigation as a Service

2.3.3.1 Description

This scenario aims at demonstrating the SDN-based DDoS mitigation framework [13] presented in deliverable D3.4, which allows an ISP and a customer networks to collaborate with each other, through their SDN controllers, to mitigate DDoS attacks. In particular, the collaboration is initiated by the customer who is expected to subscribe to the mitigation service provided by the ISP, beforehand. In fact, we have witnessed some research efforts on this on-demand service model for security functions [19], along with parallel standardizations by international organizations, such as ETSI and IETF, on *Network Functions Virtualization* [3] and *Interfaces to Network Security Function* [4], respectively.

In this scenario, the DDoS detection module is deployed at the customer network, while mitigation services are carried out at the ISP network and provided to the customer SDN controller through an interface exposed by the ISP controller. The alerts and mitigation requests generated by the customer network are sent to the ISP via this interface.

To demonstrate the effectiveness of our approach, we set up an experimental scenario in which a customer network receives video streaming flows that are disrupted by DDoS attacks. The choice of using video streaming, as the asset to protect, is due to recent studies indicating that the majority of Internet flows at peak time are video streaming flows [4]. This situation is expected to remain as such in the future.

2.3.3.2 Requirements of the Testing Environment

To demonstrate our DDoS mitigation as a service framework, we need to emulate a high-speed network environment, along with the traffic between three key players (1) video content providers; (2) Internet service providers, and; (3) customers or video service clients. In particular, the realistic traffic generated should contain both legitimate flows and attack flows. The functional components and security modules in the framework will be implemented at SDN controllers and OpenFlow switches.

2.3.3.2.1 Network For the scenario implementation, we need an ISP and a customer network environment. The ISP network in the testbed is represented by a cluster of OpenFlow switches connected to each other with high

[3] http://portal.etsi.org/portal/server.pt/community/NFV/367
[4] https://www.ietf.org/mailman/listinfo/i2nsf

speed bandwidth. The customer network in the testbed is connected to the ISP network, and is represented by having a low speed connection with the ISP network, such as to cause possible congestion in the case of an attack. The testbed platform also requires legitimate hosts and attacker hosts outside the ISP network. The legitimate hosts and attacker hosts communicate with the client by sending traffic from their respective machines.

2.3.3.2.2 Legitimate Traffic Generation The scenario requires the generation of realistic legitimate traffic. These days, around 80% percent of the Internet traffic is comprised of video streaming at peak time.

2.3.3.2.3 Attack Traffic Generation Attack traffic is required for the emulation of the scenario in the real testbed environment. The attack traffic consists mainly of DDoS flooding traffic. UDP storm, TCP SYN and ICMP flooding attack will be generated from different attack points, targeting the customer network. The generated attack traffic will cover different rates starting from 500 pps.

2.3.3.2.4 Control Infrastructure The proposed scenario provides DDoS mitigation as a service to the customers of an ISP. When the customer experiences the attack in its network, it can request the ISP to mitigate the attack by providing information on the flows which are causing the congestion in its network. The emulation of this scenario in the real testbed environment requires a control infrastructure, in which a customer can request the ISP for mitigation services. As a matter of fact, the end-to-end visibility and the logically centralized control of the SDN paradigm make it easy to manipulate flows. It is then possible to provide DDoS mitigation as an on-demand service, hosted at the ISP controller. The testbed platform requires an SDN infrastructure consisting of two controllers, one at the ISP side, one at the customer side, as well as OpenFlow switches connected through the controllers.

2.3.3.2.5 Metrics The designed framework may be evaluated in a number of ways indicative of the provided security, the performance overhead or the quality of service experienced by users of the protected network.

Precision/recall will allow to evaluate the efficiency of our mechanism over a mixture of labelled traffic in the platform. Indeed, we may reduce the packets of interest based on flows detected at the customer side, who is responsible for indicating which are the undesired flows.

Latency allows us to assess the overhead caused by the mechanism on the throughput.

Time to recover indicates the performance of our mechanism from the time an alert is raised to the time the attack is finally mitigated.

Time to rebuffer video represents, in the domain of video streaming, the pausing time during when frames are buffered to be played next. We expect this time to be greater than zero in the event of DDoS attacks impacting the video streaming traffic. The mechanism should contribute to reducing this time towards zero.

Time to start video assesses the delay caused by the DDoS attack since serious disruption to the network may postpone the start of the video. Similarly to the above metric, the DDoS mitigation mechanism should reduce this time.

Average bitrate may vary greatly as it tends to be optimized by the adaptive streaming algorithm, depending on the networking conditions. Indeed, when network conditions degrade due to congestion, possibly caused by a DDoS attack, the video level is degraded in order to prevent the video from stopping (in order to rebuffer). When the bandwidth increases, the bitrate increases as well. From the perspective of the user, it also contributes to a better quality of experience, since the overall quality of the streamed video will be better.

Our mechanism can also be evaluated from the user perspective thanks to these metrics.

3 Botnet Introspection

Botnets may serve as a powerful arsenal to launch various massive attacks, such as DDoS and spamming. However, completely eradicating botnets is mission impossible in practice due to both technical and social challenges. This use case is dedicated to leveraging and correlating partial observations at distinct layers for constructing a holistic framework to analyze botnet activities with more certainties, ultimately reducing response latency and yielding more effective mitigation mechanisms.

3.1 Description

The malicious effect of botnet spreads in various ways, e.g., distributing malicious software, generating unpleasant traffic, and stealing personal information. The behavior of botnets however converges into a single model: a communication between central or distributed control hosts (Command and Control servers, or C&C servers) and compromised hosts. A malicious party employs C&C servers and directs compromised hosts by sending commands to behave as they want. Understanding such commands and the behavior afterwards are thus important to circumvent their activities.

We focus particularly on two behaviors of botnets in this use case analysis. The first one is on the initial rendezvous of communication between bot programs, which run on compromised hosts, and the C&C server, as well as the investigation of such traffic leveraging threat analysis from passive measurement. The intelligence obtained by workpackage 2's outcomes contributes to identifying the malicious traffic generated by controlled bot programs in a sandbox environment or a synthetic tool. The second focus is on the behavior of traffic generated by decentralized, peer-to-peer bot programs at compromised hosts. With the clustering of the live traffic collected by NetFlow probes, the malicious traffic shall be detected in a lightweight manner within a short amount of time.

Every scenario is able to generate indicators of threats which are later used by defense mechanisms. Although scenarios can be extended to mitigate such malicious activities carried out by botnets, we do not focus on the countermeasure or mitigation in this use case analysis. Mitigation itself will be covered by the *Malware Campaign Mitigation* use case.

3.2 Metrics

This section describes the metrics used in this use case.

3.2.1 General Metrics

3.2.1.1 Detection accuracy

Type: Quantitative

Definition: The value of successful detection for malicious traffic or compromised host. In other words,

$(TruePositive + TrueNegative)/(TruePositive + FalsePositive + FalseNegative + TrueNegative)$

Score: %

Measurement: depends on individual scenario

3.2.1.2 Detection precision

Type: Quantitative

Definition: The number of true positive detection out of total positive detection for malicious traffic or compromised host. In other words,

$(TruePositive)/(TruePositive + FalsePositive)$

Score: %

Measurement: depends on individual scenario

3.2.1.3 Detection recall

Type: Quantitative

Definition: How many out of the "actual bad" objects are detected.

$(TruePositive)/(TruePositive + FalseNegative)$

Score: %

Measurement: depends on individual scenario

3.2.1.4 Delay

Type: Quantitative

Definition: The delay of legitimate traffic induced by detectors. The difference is measured between two situations: when the detector is enabled, and when it is disabled.

Score: sec/msec/usec

Measurement: traffic monitoring is used to compare between the system with the detector and without.

3.2.2 Scenario-specific Metrics

3.2.2.1 Collected information at decoy server

Type: Quantitative

Definition: The amount of commands collected at the decoy server.

Score: [n] (the number of commands)

Measurement: pcap files and log files measured at decoy server

3.2.2.2 Detection time

Type: Quantitative

Definition: Time between infection of a client/network and detection of the infection

Score: minutes/hours/days

Measurement: the time from the start of the infection until infection has been accurately identified.

Table 3.1: Metrics to be collected for botnet introspection scenarios

General metrics			
Metric	Description	Units	Remark
Detection accuracy	accuracy of detection	%	
Detection precision	how many detected items are "actual bad"?	%	
Detection recall	how many "actual bad" items were detected?	%	
Delay	the delay imposed on legitimate traffic	second	
Scenario-specific metrics			
Collected information at decoy server	the amount of collected commands	[n]	
Detection time	Time between infection of a client/network and detection of the infection	minutes, hours, days	

3.3 Scenarios

3.3.1 C&C Commands Collection through Decoy

3.3.1.1 Description

A centralized botnet relies on a C&C server to maintain and control each botnet entity. Attackers operate botnets via C&C servers. Investigating communication between bots and C&C servers represents a significant portion of botnet introspection. This scenario investigates traffic between live bot clients and their C&C server using a decoy server.

This scenario uses the Request Policy Zone (RPZ) mechanism to redirect traffic generated by bot-infected clients and directed to their C&C server.

RPZ is a mechanism implemented in the BIND9 nameserver software. RPZ allows the operator to modify responses to queries: it uses a simple rewrite ruleset. An operator could replace IP addresses linked to botnet's hostnames with some IP address controlled by him (i.e., the decoy server).

In this scenario, we use a malicious domain names list created by our C&C domain name detection method developed in workpackage 2. This method detects malicious domain names by applying a machine learning algorithm to observed DNS data – dynamic analysis of live bot(s), DNS traffic on a cache server, and DNS traffic on a root nameserver.

The nameserver rewrites IP addresses of listed hostnames in order to redirect malicious traffic to the decoy server that inspects connections to malicious hosts.

3.3.1.2 Requirements of the Testing Environment

This scenario reveals feasibility of our developed botnet introspection framework that targets botnet clients in a network operational domain.

Hence, the testbed platform simulates one independent and autonomous network service system - it consists of client hosts, a router/gateway, and a DNS cache server.

3.3.1.2.1 Network Our detection technology uses dynamic analysis of a live bot client. In general, botnet clients need connectivity to start their activities; if a botnet client cannot detect any connectivity to the Internet, they stop their action. Specifically, we have to provide limited connectivity to intentionally malware-infected host that only allows a critical communication such as a few DNS queries and a few access to major sites used for checking connectivity.

Therefore, this scenario requires segregated, controlled network for dynamic botnet client analysis.

3.3.1.2.2 Data Source Our detection method requires the following data sources:

- Queries from a live bot in a sandbox
  This approach uses dynamic analysis approach to identify candidates of malicious domain names by creating a virtual machine and intentionally infecting it with a botnet client. We use observed DNS traffic from the infected VM for candidates of malicious domain names.

- DNS Query traffic on a DNS cache server
  DNS query log is used for reinforcement learning.

- DNS Query log from root nameserver
  The detection method needs a query log for authoritative nameserver that is queried by widely-distributed client. The query log from root nameserver satisfies these requirements.

The detailed description of the detection method can be found in D2.1.

3.3.1.2.3 Traffic Generation We employ two types of traffic data: one of them is DNS traffic data that is produced by live bot clients in a sandbox. We use this data for proof of concept about our framework, in addition to

the observed traffic in a real network. We use this traffic to demonstrate feasibility in the live environment.

3.3.1.2.4 System Components This scenario consists of the following components:

- Decoy C&C server
  Decoy C&C server listens and records any connection from malicious hosts for investigating botnet clients.
- Nameserver configured with RPZ
  The DNS cache server, configured with RPZ, that responds to client DNS query.
- Hypervisor with Cuckoo Sandbox[1]
  A virtual machine hypervisor that runs infected VMs.
- bot-infected virtual machine(s)
  Hosts that execute botnet client binary. We use two botnet clients in this scenario: one of them is used for dynamic analysis as a part of the detection system and the other client is used to emulate a victim host. In order to evaluate the general versatility of our framework, botnet client binaries are not necessarily the same between these clients.

3.3.1.2.5 Metrics The precision of the botnet introspection system will be evaluated by the following metrics: detection accuracy, precision, and recall.

Accuracy/Recall/Precision to measure the accuracy of the detecting mechanism. Ideally, the system has to detect actual bot-infected hosts without false negatives, and also should not wrongly regard a legitimate host as malicious. These parameters will be calculated by passing a combination of well-known malicious domain names and legitimate domain names to the system.

Collected information at decoy server indicates the amount of information about botnet activity that is collected by the decoy server.

DNS query per second/DNS resolution time indicates the overhead of this platform. These parameters will be measured by querying a combination of malicious domain names and legitimate domain names to the system.

[1] http://cuckoosandbox.org/

3.3.1.3 Sequence Diagram

Figure 3.1 illusrates both the workflow and the topology of this scenario.

1. The DNS analysis module creates a malicious domain list using DNS datasets (① prepare).
2. A created RPZ domain name list will be sent to the RPZ DNS cache server (② install).
3. The botnet client sends a DNS query for the C&C server's domain name that is recorded in the RPZ domain name list.
4. The DNS cache server responds with a modified DNS response to the botnet client.
5. The botnet client is redirected to the decoy server.
6. The decoy server accepts connections from the botnet client. It records these transactions and commands.

3.3.1.4 Topology Diagram

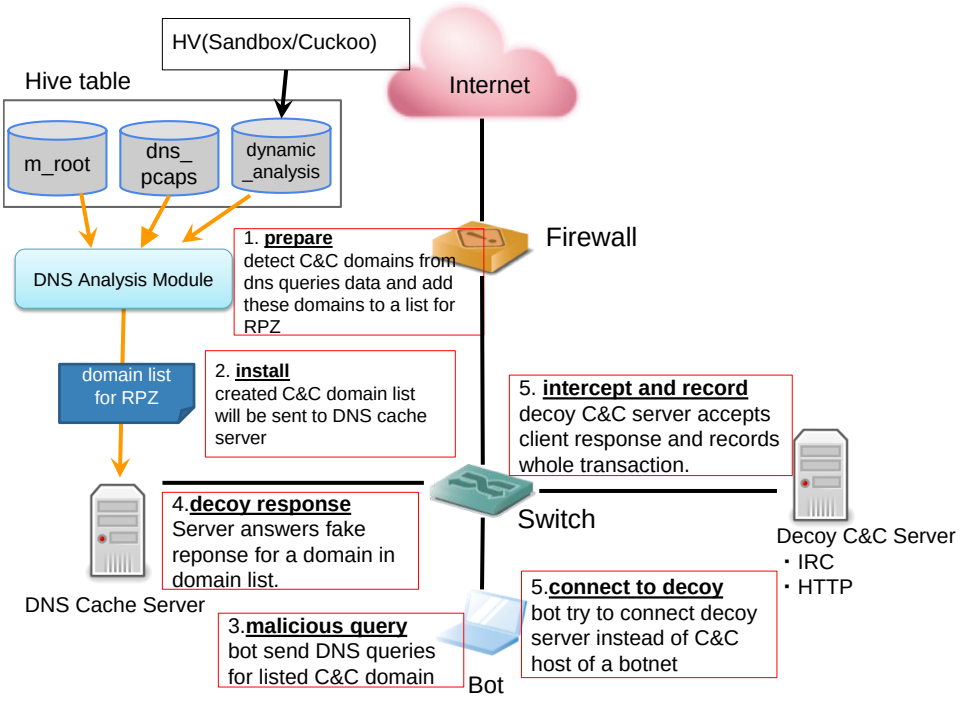

Figure 3.1: Topology of C&C command collection through decoy.

3.3.2 DGA Filtering

3.3.2.1 Description

This scenario presents a filtering mechanism for malicious DNS queries that targets communications within botnets. In general, the botnet clients try to communicate with C&C servers. The servers control the bots and initiate attacks by sending control messages to the hosts. Then the bots have to establish communication channels with the control servers. However, the servers' addresses are not directly encoded on the client malware, in order to avoid easy detection by security software and probes. Alternatively, the client malware adopts dynamical domain generation to communicate with C&C servers. Most domain generation mechanisms rely on Domain Generation Algorithms (DGA). For example, some malware, such as the ZeuS trojan horse, generate domains based on dates to frequently change the domains hosting their C&C server.

The proposed mechanism mitigates the queries before the queries are resolved by DNS servers to block initial communications. Entire DNS queries are intercepted by the mitigation mechanism on edge networks. The mechanism blocks the queries which are included in a blacklist shared by NECOMAtter, which is an external knowledge base developed in workpackage 3 (cf. deliverable D3.2). The legitimate queries are forwarded to legitimate resolvers by the mechanism.

3.3.2.2 Requirements of the Testing Environment

3.3.2.2.1 External knowledge base The knowledge base provides a list of malicious domains for the mitigator.

3.3.2.2.2 Information sharing platform The platform used to share the malicious domains between the detector and the mitigator.

3.3.2.2.3 Mitigator The host used to filter DNS queries based on blacklisted domains. In the mean time, the host should forward other legitimate queries to DNS resolvers.

3.3.2.2.4 Victim host Hosts operating as member nodes of a botnet. These hosts generate DGA-based DNS queries, that imitate an initial communication among bots and C&C servers.

3.3.2.2.5 Legitimate host The host sends DNS queries which contain legitimate domains.

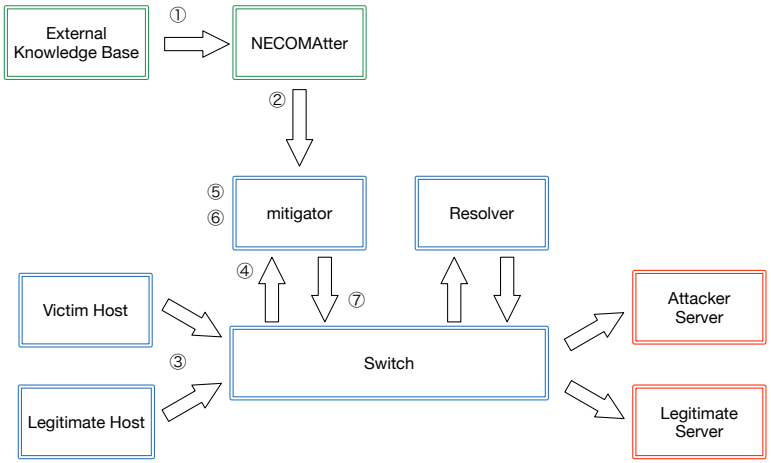

Figure 3.2: Scenario sequence of a filtering mechanism for DGA queries

3.3.2.2.6 Metrics

Recall/Precision to measure the accuracy of the filtering mechanism. The mechanism filters DNS queries based on domain lists. There is a possibility that the mechanism blocks legitimate queries and allows passage of malicious queries.

Latency to measure the latency of legitimate DNS queries with and without the filtering mechanism. In the scenario, the filtering mechanism intercepts both legitimate and malicious DNS queries from user hosts to a DNS server. The legitimate queries could be delayed due to the mechanism. The delay should be short to avoid impact on the legitimate queries.

3.3.2.3 Sequence Diagram

Figure 3.2 shows the sequence diagram of this scenario. The details are described below.

1. The external knowledge base posts a list of malicious domains on NECOMAtter.

2. The mitigator node acquires the list from NECOMAtter. Then the mitigator generates filtering rules based on the list and enforces the rules itself.

3. The victim node generates malicious DNS queries using DGA algorithms. In addition, the legitimate node sends legitimate DNS queries.

4. Both packets of the queries are forwarded to the mitigator.

5. The mitigator classifies incoming DNS packets based on the defined rules.

6. If the `QNAME` field of a received query is matches a rule, the query packet is dropped by the mitigator.

7. Other unmatched queries are forwarded to the DNS resolver.

3.3.3 Bot Detection based on Traffic Anomalies

3.3.3.1 Description

This scenario aims at investigating the behavior of a bot-infected enterprise network infrastructure, where the infection is spreading internally. It will concentrate on analyzing internal network traffic, in NetFlow format, looking at the inter-machine communication, trying to distinguish different patterns and identify anomalies in outgoing and incoming traffic for nodes in the network. The scenario will focus on detection of decentralized, peer-to-peer botnets, where infection is spreading among machines within the network. Because the infection will be spreading within a certain time frame, the *Detection time* metric is also relevant for this scenario, and may give valuable insights on botnets behavior.

The network infrastructure design, as well as the generated traffic patterns, will be based on real enterprise network infrastructure and traffic, although due to privacy issues, several elements of the network, including the generated traffic, will be anonymized or modified.

The proposed detection mechanism is able to cluster traffic nodes based on the analysis of NetFlow data which allows much lightweight, nearly real-time analysis. Later, a sensor monitors the network communication patterns between two or more machines, located in the same network, and based on the PageRank [10] algorithm conducts a classification of the machines.

3.3.3.2 Requirements of the Testing Environment

The scenario requires an isolated environment, not to interfere with regular network traffic, hence, a simulated, sandboxed, enterprise network would be desired.

3.3.3.2.1 Network The minimum requirements to properly simulate the scenario are listed below:

Communications node to coordinate network traffic, e.g., a router.

NetFlow traffic probe on the communications node required to capture network traffic between the network machines. It has to be capable of capturing and exporting the traffic in NetFlow format.

Machine dedicated to running the analysis i.e., the NetFlow analysis module.

Machines within the network at least, two machines connected through the communication node.

In order to make the scenario more realistic, providing enterprise services for the internal machines through the simulation of internal DNS and Web Servers is also foreseen. Since those infrastructure elements generate specific traffic patterns, they will have an influence on the results, which would be an additional challenge for the sensors in a scenario where the servers are not known beforehand. This will show how the sensors perform in different scenarios measuring parameters like accuracy, recall and precision.

3.3.3.2.2 Traffic Generation The generated traffic will be based on real enterprise network traffic, where each machine in the simulation will resemble, to the highest extent possible, real machines belonging to an enterprise network. The messages sent between the machines in the network will imitate a regular working day in an enterprise, i.e., including using enterprise services and DNS queries from regular employees machines within an enterprise. Sequentially, the network will be infected with bots injecting malicious traffic into the network.

3.3.3.2.3 Metrics The sensors will be assessed in terms of performance by looking at common metrics like accuracy, recall and precision, as well as scenario-specific metrics, such as the detection time.

Accuracy, recall and precision are standard performance parameters that measure the efficiency of classifier systems. The measurement is carried out by comparing the classification, made by the sensor, with the true labels of the machines in the network, where the positives are infected machines and negatives are machines that are not part of the botnet. Taking into consideration an enterprise network, the aforementioned parameters are crucial, since they may have an impact on the actual productivity and availability of services provided by the enterprise, where a false positive could disturb a normal working schedule of employees or enterprise servers.

Detection time is the time elapsed between the first bot activation and the actual detection of the botnet by the sensors. Detection time will allow to measure how much time is needed to detect an infected network,

measuring the time from the beginning of the infection until accurate identification of the infection. Additionally, it will also allow to estimate how much the malicious traffic disturbs the normal operation of the network by e.g., measuring the delay by comparing the broadcast of a message in the network during an infection and in a normal, non-infected state, before the infection is detected.

4 Smartphone User Protection

The attack surface of today's cyberspace has been significantly enlarged by the rapid increase in smartphone usage and the proliferation of diverse mobile applications, which introduce a large variety of zero-day vulnerabilities. In this use case, we aim at validating how the resilience at the human and social level could be improved by employing our new approaches developed in previous workpackages. Note that if subject-based studies are required, the malicious websites visited by the participants should not be publicized in order to avoid information leakage and malware infection.

4.1 Description

There are numerous motivations for cyber attacks targeted at smartphones. One fact is that the number of smartphone users is large and it keeps increasing sharply. According to BI Intelligence, at the end of 2013, 6% of the global population owns a tablet and 20% owns a PC, while 22% owns a smartphone[1]. Another fact is that smartphones often hold much more personal information than PCs do, including the detailed records of users' contacts and SMS history, as well as sensitive account information regarding banking, emails, social networks, etc.

Mobile malware is one of the most significant cyber threats on smartphones. According to Schmidt et al. [16], smartphones started being the targets of malware since June 2004. As of January 2014, roughly 700,000 of cumulated Android malware samples have been observed, according to a report published by Sophos [18]. One good example is iBanking, a criminal software targeting Android terminals [12]. Specifically, iBanking Mobile Bot is controlled via HTTP or SMS, and it allows bot herders to steal personal

[1] http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10

information by reading SMS messages, intercepting incoming calls, and obtaining sensitive files such as the contact list. It also has a function of phone fraud, which allows a botmaster to earn money by calling premium rate telephone service while charging the victim an expensive toll fee. Another function of iBanking is to record audio using the phone's microphone. Note that there are many services that require users to input credentials such as credit card and/or PIN number over tone signaling, and this function can recognize the context of typing by the key tones [15]. Nowadays, smartphones are also used in DDoS attacks. For example, Android DDoS Origin[2] is a malware controlled via SMS messages, which specify the victim's host and port number. Indeed, it makes smartphones generate anomalous traffic.

In addition, smartphone users are vulnerable to phishing attacks, since the user interfaces for smartphones are constrained by their small screens. In legacy PCs, symbols indicating trust have been developed for a long time. For example, web browsers may display a padlock icon containing a valid server certificate, and the green color is used in the address bar when a server is equipped with an extended validation (EV) certificate. The users can therefore be aware of trust information. So improving awareness about security indicators is important to improve user interfaces and make them more trustworthy. However, for smartphone users, there are quite few indicators for security awareness.

4.2 Metrics

This section defines a set of metrics for evaluating the smartphone protection mechanisms, as well as the requirements for testing environment.

Effectiveness The defense mechanisms must achieve high detection accuracy. Apparently, user safety would be compromised if the defense system labeled malicious entities as benign. Conversely, users would also complain if the defense system labeled benign entities as malicious because of the interruption caused by the system.

Overhead While the primary design objective of our defense mechanisms is to effectively prevent, detect and mitigate cyber threats and recover any damages, we also expect them to be lightweight and work in resource efficient ways.

In the following, we specifically define the metrics, appropriate to our use case, that allow to measure the *effectiveness* and the *overhead* of our mechanisms. In addition to those general metrics, some particular metrics may be defined to evaluate the outcomes of scenario-specific models, characteristics, and/or situations.

[2] http://news.drweb.com/show/?i=3191&lng=en

Table 4.1: Confusion matrix for effectiveness metrics

	Label as Threat	Label as Benign
Actual Threat	True Positive (TP)	False Negative (FN)
Actual Benign	False Positive (FP)	True Negative (TN)

4.2.1 General Effectiveness Metrics

The metrics with respect to *effectiveness* are defined as follows. Note that Table 4.1 summarizes the factors for calculating each metric.

4.2.1.1 Detection accuracy

Type: Quantitative

Definition: The value of successful detection for malicious events. It can be calculated by $\frac{(TP+TN)}{(TP+TN+FP+FN)}$ (the higher the better).

Score: %

Measurement: depends on individual scenario.

4.2.1.2 Detection precision

Type: Quantitative

Definition: The number of true positive detection out of total positive detection for malicious traffic or compromised host. It can be calculated by $\frac{TP}{(TP+FP)}$ (the higher the better).

Score: %

Measurement: depends on individual scenario.

4.2.1.3 Detection recall

Type: Quantitative

Definition: How much of the "actual bad" objects can it detect? It can be calculated by $\frac{TP}{(TP+FN)}$ (the higher the better).

Score: %

Measurement: depends on individual scenario.

4.2.1.4 Time to Recover

Type: Quantitative

Definition: The duration between the time when the services were affected under attack and the time when the attack is mitigated (the lower the better).

Score: minutes/hours/days

Measurement: at the defense system.

4.2.2 General Overhead Metrics

Since resilient cyberdefense aims at recovering damages resulting from attacks, a restoration procedure should be performed in a short time period. In addition to *effectiveness* metrics previously defined, the protection mechanisms should be lightweight considering the limited resource of smartphones. For example, the user experience might be penalized if the mechanism requires heavy CPU load and/or memory consumption.

4.2.2.1 Delay

Type: Quantitative

Definition: Time between the time when defense procedure began and the time when it completed (the lower the better).

Score: minutes/hours/days

Measurement: at defense system.

4.2.2.2 Average CPU Load

Type: Quantitative

Definition: Average of the duration to load the system (the lower the better).

Score: score

Measurement: at the defense system.

4.2.2.3 Memory consumption

Type:	Quantitative
Definition:	The sum of all the memory consumed by the threads of the defense mechanism (the lower the better).
Score:	bytes
Measurement:	at the defense system.

4.3 Scenarios

4.3.1 Phishing Protection

4.3.1.1 Description

This scenario demonstrates a personalized phishing protection mechanism. Here personalization means that the mechanism can be customized per the needs of end user. The mechanism involves the observation of the user's awareness when visiting trustworthy or phishing websites, and the differentiation of countermeasures according to the skill of users. In particular, the observation of awareness is a fundamental concept for determining the skills of a user. For example, based on our analysis results [8], novices usually fail to make correct decisions, since they are mostly attracted by web contents (which always appear to be similar between phishing sites and legitimate sites), rather than URL or SSL indicators. By contrast, an expert tends to evaluate the site's URL and/or the browser's SSL indicator rather than the contents of the web page to determine the credibility of the sites.

The system checks the level of awareness with respect to the address bar for each participant. By interacting with the eye-tracking device, the system will determine if the participant is a novice, an expert or another type of participant. For an expert participant, the defense system employs ATOS's high precision analysis modules. For a novice participant, the defense system uses UTokyo's machine learning-based analysis modules. These modules were described in deliverable D2.1.

The researcher needs to conduct participant-based experiments. The participants were categorized on the basis of their eye movement, and were provided phishing protection mechanism for them. The defense should achieve high detection accuracy and perform with low latency. We expect that the defense would not penalize the participants' user experience, therefore, the test will ask the participant to rate his/her satisfaction as well as measuring the overhead of the defense system. We also expect the participants to acquire the habit of checking the address bar to prevent phishing.

4.3.1.2 Requirements of the Testing Environment

The scenario resembles phishing IQ test, as described in deliverable D1.3: it presents participants with legitimate sites and phishing sites. The main difference is that this scenario employs a smartphone and its embedded defense system to prevent the participants from phishing.

To demonstrate our phishing prevention for smartphone users, we need to prepare typical web browsing environments. The test environment consists of web clients, web servers, and a defense system based on an eye-tracking device, as well as a network infrastructure to connect the components.

4.3.1.2.1 Web clients Due to the nature of the scenario, we need to setup smartphones that a participant will use. It should run Android OS or Windows 8.1, and be configured to browse legitimate or phishing websites.

The policy decision points are the browser and its extention. The browser therefore might be extensible to interact with the defense systems, such as eye-tracking camera and analysis modules implemented as NECOMAtter bots.

4.3.1.2.2 Phishing servers Phishing servers host phishing content for web clients. Since the lifetime of phishing sites tend to be short, we will crawl phishing contents, or store, and reproduce it in the test environment. To avoid information leakage, these server should be isolated from the Internet.

4.3.1.2.3 Legitimate servers The legitimate websites used in the scenarios will be actual websites in the wild due to the difficulty of preparing EV-SSL certificates in the testbed.

4.3.1.2.4 Eye-Tracking camera A camera for recognizing users' eye activity. Based on our previous analysis [8], we will employ eye-movement-based observations since eye-tracking allows for the direct observation of the user's behavior. It should be easily applicable to people. No-contact devices might be preferable.

4.3.1.2.5 NECOMAtter It is used to communicate between EyeBit (the defense mechanism) and the analysis modules.

4.3.1.2.6 Network Network accessibility is necessary for the interaction of the testbed components.

4.3.1.2.7 Metrics This section describes the required metrics for the scenario. We will assess the success of our mechanism based on *effectiveness*, *overhead* and scenario specific metrics described as follows.

Detection accuracy can be used for measuring the effectiveness of the defense system. In this scenario, our primary motivation is to protect smartphone users from phishing, which can be assessed by the detection accuracy of the defense system.

Detection precision/recall can be also used. We will use the different analysis modules for different types of participant. The ATOS's analysis module focuses on high precision, which is not the case for UT's analysis module. It might be helpful to distinguish the effectiveness of the different modules.

Delay can represent how long does a participant have to wait until he/she makes decisions. There is a trade-off relationship between security and usability, and the defense system usually penalize a participant's user experience. We regard the delay as an objective metric for the loss of usability.

User experience is a scenario-specific metric. This is a qualitative variable, collected through a questionnaire. The participants assess the defense mechanism according to a five-grade evaluation system (e.g., outstanding, excellent, good, fair, and poor).

We regard it as a subjective metric for the loss of usability. The damage to the user experience may differ depending on the involved defense mechanism since the defense is differentiated based on the type of participant.

Educational Effect (optional) is also a scenario-specific metric. This is a quantitative variable, measured via a follow-up study. There might be an educational effect on the participant who may acquire phishing prevention habits through the participant-based experiment. The follow-up study evaluates if the habits still remain for the participant after the experiment period.

4.3.1.3 Sequence Diagram

The sequence diagram of this scenario was described in deliverable D3.4.

4.3.2 Smartphone Firewall

4.3.2.1 Description

This scenario demonstrates a smartphone firewall which aims at offloading the smartphone firewalling function to network switching devices. It is

due to the excessive energy consumption required to deploy cyberdefense mechanims at the smartphone itself. Initial design was described in deliverable D3.4. We employ OpenFlow-capable wireless access points (APs). Since OpenFlow provides powerful traffic control schemes, it facilitates the implementation of URL filtering based on the packet payload, as well as packet filtering based on the header information of the network and transport protocols such as IP address, and TCP/UDP port numbers.

In this scenario, the functions of the smartphone firewall are as follows.

- *Preventing a smartphone from joining a DDoS attack.* In the case where a smartphone device is not the target of a DDoS attack, it may however be infected with a trojan horse, prompting it to send data packets to a specified host whenever it receives a DDoS attack command.

  It should be noted that the scope of DDoS mitigation is usually at the infrastructure layer rather than at the endpoints, even when the assets to protect are smartphones. However, a major principle of DDoS protection stipulates that the mechanism should filter DDoS traffic as close to the attacker as possible.

- *Blocking the access to malicious websites.* Since the screen size of the smartphone tends to be smaller than the PC's or laptop's, it is uneasy for users to recognize URL or security information shown by smartphone browsers. Additional protection mechanisms might be necessary.

- *Blocking the propagation of malware from compromized smartphones.* The smartphone firewall has the ability to prevent infected smartphones from attacking neighboring smartphones in some types of network. For example, carrier networks tend to prohibit that a smartphone connects to other smartphones in the same network. However, in future networks, there is a possibility that carrier networks allow a smartphone to connect to other smartphones (e.g., virtual access point as proposed in IETF NVO3 WG[3]). In such case, a smartphone firewall will be helpful to thwart the attacks.

4.3.2.2 Requirements of the Testing Environment

This scenario is similar to the scenario for *SDN-based DDoS Mitigation* scenario described in Section 2.3.2, which makes use of an OpenFlow controller and switches to mitigate DDoS attacks. The key difference is that the scenario in Section 2.3.2 protects servers from DDoS, whereas this scenario aims at preventing smartphones from joining DDoS attacks.

To demonstrate our smartphone firewall, we have to build an experimental environment described as follows.

[3] https://datatracker.ietf.org/wg/nvo3/documents/

4.3.2.2.1 A smartphone device In this scenario, a smartphone device must connect only to the OpenFlow-capable APs complying with IEEE 802.11 standards. It must not connect to another network, such as a cellular network provided by a mobile carrier, since it is required that it may be able to join a DDoS attack or to visit malicious websites. In the scenario for DDoS prevention, a (pseudo) bot is required to send numerous packets towards the victim.

4.3.2.2.2 OpenFlow-capable APs (Mitigation Switch) The policy enforcement point is a wireless access point (AP) to which the smartphone connects. It should interact with an OpenFlow Controller.

4.3.2.2.3 OpenFlow controller (Mitigator) Similarly to the *SDN-based DDoS Mitigation* scenario, the controller is required to have a RESTful API that will allow for injecting filtering rules that will filter packets on the APs. It should interact with the PDP through NECOMAtter.

4.3.2.2.4 NECOMAtter and its bots NECOMAtter is a platform used to establish communication between the OpenFlow controller and some NECOMAtter bots (or modules). In the NECOMAtter system, these bots are required to run analysis modules (e.g., PDP bots).

4.3.2.2.5 (Pseudo) bot master A server that launches DDoS attacks and lets smartphones join the attack. It must be isolated from the Internet to avoid attacks to servers in the wild.

4.3.2.2.6 (Pseudo) victim node A target of the DDoS attack. It must be also isolated from the Internet, in order to prevent sending backscatter packets.

4.3.2.2.7 Legitimate servers We also use several legitimate servers in order to assess the delay for accessing the legitimate websites.

4.3.2.2.8 (Pseudo) malicious web server Malicious web servers that host malicious web contents for unsusupecting web clients. The served contents include phishing, drive-by-download, or other types of malicious modus operandi. To avoid information leakage, these servers should be isolated from the Internet.

4.3.2.2.9 Metrics This section describes the required metrics for the scenario. We will use *overhead* and scenario-specific metrics as follows.

Delay allows to assess the function of blocking the access to malicious websites, which may suffer from delays when the smartphone firewall communicates with the OpenFlow controller.

Time to recover indicates the performance of our mechanism from the time an alert is raised to the time the attack is finally mitigated.

Energy consumption is a scenario-specific metric. The use of the smartphone battery is monitored during a specific time period. It can be measured by the smartphone utility tools, or can be also estimated by the number of generated packets by the smartphone. The battery of the smartphone will be drained by the attack due to numerous packets. Measuremeant is performed continuously until the end of the mitigation. We will measure and compare the battery consumption with and without the smartphone firewall.

4.3.2.3 Sequence Diagram

The sequence diagram of this scenario was described in deliverable D3.4.

4.3.3 Drive-by-Download Prevention

4.3.3.1 Description

The scenario aims at demonstrating how to prevent drive-by-download attacks on a user's browser. Drive-by-download attacks use obfuscated malicious JavaScript code on a web server to redirect users to malware distribution servers. A user who accesses a landing server downloads malicious code and executes it in the context of the browser. The malicious code exploits a vulnerability that will allow to redirect the browser's execution flow. Hijacking the browser's execution usually leads to downloading a malware hosted at a malware distribution server. Finally, the malware is injected in the user's host, where it runs with the privileges of the browser, waiting for some commands from the attacker.

In this scenario, we demonstrate a blocking mechanism which performs on top of a browser and terminates an access to a distribution server. The defense mechanism works as a browser extension on the Chrome browser. When the browser downloads an HTML file with obfuscated JavaScript code, the extension de-obfuscates the code. If the de-obfuscated code includes access requests to domains different from the domain of the landing site, the extension blocks the access and raises a warning to the user.

4.3.3.2 Requirements of the Testing Environment

This scenario requires the emulation of different services including malicious web pages controlled by the attacker (a landing page and a malware distribution page), an information sharing service (NECOMAtter), as well as a victim node. The victim host has to install the browser extension to block drive-by-download attacks on the Chrome browser.

4.3.3.2.1 Network There are no specific network requirements in this scenario. The scenario can be emulated on a single segment network, because the scenario entities do no use any L2/L3 information.

4.3.3.2.2 Attacker Hosts At least two attacker servers are required for this scenario. One server is used to host a landing site. The landing site should respond with a web page containing a malicious JavaScript code that will ultimately lead the user's browser to a malware distribution site. Another one is for hosting the malware distribution site. This server is supposed to distribute malware to the user's browser. However, since we do not need actual malware in this scenario, it is just a downloadable file.

4.3.3.2.3 Victim Host The host has to be able to operate Chrome browser. In addition, the host should have a display to show the browser screen.

4.3.3.2.4 Information Sharing Platform This platform is used to share malicious URLs among users. Therefore, the platform should have an interface for posting the URLs in text format.

4.3.3.2.5 Metrics

Accuracy to measure how the mechanism accurately blocks access to malware download servers. The mechanism filters redirections based on domain names which were concealed in the landing page's web contents. Conversely, some legitimate site could be blocked by the mechanism.

Latency to measure the latency of legitimate web access with and without the mechanism. The mechanism works on a user's browser and analyzes JavaScript code, downloaded from web sites, even if they turn out to be legitimate. It may involve some delay for any legitimate web site accessed by the browser.

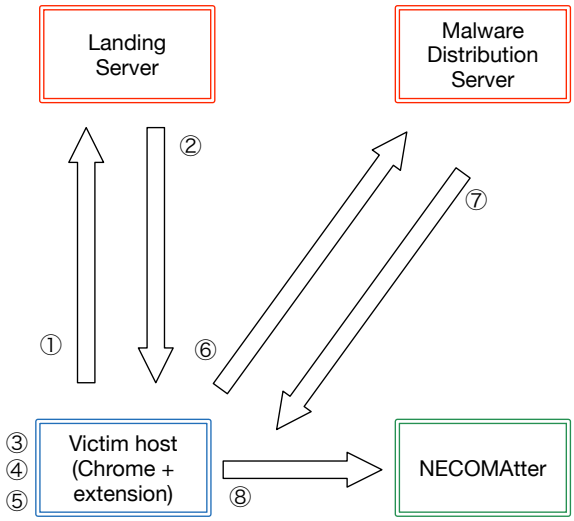

Figure 4.1: Scenario workflow for drive-by-download prevention

4.3.3.3 Sequence Diagram

This subsection describes the workflow of the scenario. Figure 4.1 shows the sequence diagram of the scenario. We explain the sequence as follows:

1. The Victim user's device opens the URL of the malicious server.
2. The browser downloads web contents which contains obfuscated malicious JavaScript code.
3. The browser executes the code.
4. The browser extension analyzes the code and blocks access to the download server.
5. The extension displays an alert for the browser's user.
6. The extension posts the URL of the malicious site and the download site on NECOMAtter, which is an information sharing platform developed during this project.
7. If the extension is disabled, the victim host accesses the download server.
8. The malware is downloaded to the victim host and the host executes the malware.

4.3.4 SMS Fraud Protection

The proliferation of smartphones has led to the growing prevalence of mobile malware. Especially, the most popular mobile platforms, such as Android, attract an increasing number of attackers, that try to develop malicious applications and distribute them through the marketplaces to the end users. One category of such applications is financial malware which aims at causing financial loss to the infected end users. The openness of the Android marketplace makes it a hot target of such kind of malware attacks. These attacks on Android take the form of applications attempting to charge the users without their knowledge by sending SMS messages or initiating calls to premium rate numbers, and are known as *premium-rate dialers* and *SMS fraudsters*. These kind of threats clearly call for better next-generation anti-malware solutions. What follows is a description of how these threats can impact mobile users and a solution we developed in order to mitigate such attacks in real time.

4.3.4.1 Description

Premium-rate dialers and SMS fraudsters.

As Lookout mentions in its *State of Mobile Security 2012* report [7], there has been a notable rise in the prevalence of financial malware which aims at gaining profit from unaware mobile users by unlawfully billing them. This class of malware has different forms. For example, SMS fraudsters and premium dialers are kinds of malware falling into this category. SMS fraudsters exploit the premium-rate text messages (premium SMS messages), that allow people to conduct online payments easily by charging an amount of money in their phone bill directly. An overview of how a legitimate pre-

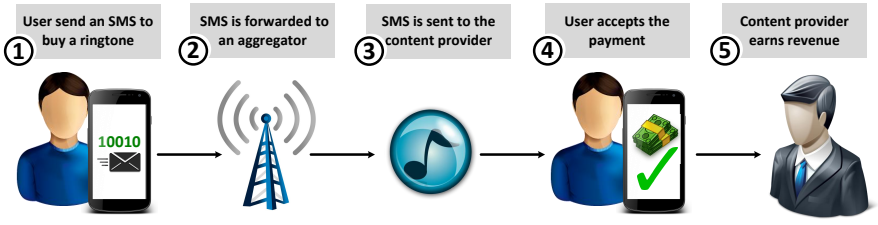

Figure 4.2: Overview of how legitimate premium SMS works.

mium SMS service works is shown in Figure 4.2. A user sends an SMS in order to charge a mobile service (e.g., to make a payment) to her phone account (step ①). Then, a wireless provider will forward the SMS to an aggregator (step ②). The aggregator, in turn, sends the SMS order to the respective service provider, which asks the user, through another message,

to confirm the payment she submitted (step ③). Once the user accepts the payment, she receives the requested service and her phone bill is charged (step ④). Then, the corresponding provider pays the aggregator and the aggregator, in turn, pays the respective service provider.

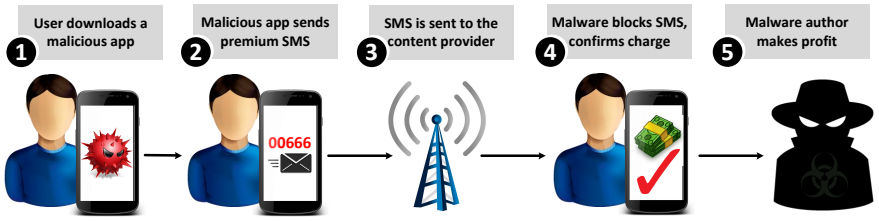

Figure 4.3: Overview of how premium SMS fraud works.

This process can be exploited by an attacker through a mobile malware in order to gain profit. An overview of how a malicious premium SMS service works is depicted in Figure 4.3. First, the user accidentally downloads and installs on her phone a malicious application (e.g., an SMS fraudster app) (step ❶). Once installed, the malicious app will send premium SMS without user's awareness (step ❷). Then, the process continues as in the case of the legitimate premium SMS service resulting in a message sent to a user to confirm the payment (step ❸). Afterwards, the malicious app will intercept the message and accept the payment, without user's awareness (step ❹). Through this, the money of the payment will go to the malicious writer (step ❺).

We presented how SMS fraudsters work. Premium-rate dialers function in a very similar manner, using phone calls instead of SMS in order to cause financial loss to smartphone users.

Proposed Countermeasure.

In order to protect smartphone users against this kind of threats, we have implemented a system that intercepts all the SMS messages and phone calls attempts in real time, leaving the user to decide whether to send a message or perform a call. Our scheme requires a CAPTCHA [1] validation before the user proceeds to an action so as to prevent malware from simulating an unexpected user behavior. In order for our system to be more usable, it is equipped with two protected structures, a blacklist and a whitelist, so that the user can add there her preferences about the numbers she wants to block or trust, allowing the system to remember those preferences in the future.

4.3.4.2 Requirements of the Testing Environment

In this section we describe the technical requirements for the scenario. We need a mobile device and some hosts for the demonstration as following.

4.3.4.2.1 Network

This scenario requires a set of nodes connected to the Internet. These nodes include the legitimate node, that is a handheld device or a device emulator, a malicious node on which an attacker has access and a host running the App Marketplace software. Because the Marketplace will host a set of real-world malicious applications and these apps will also be installed and run in the legitimate node, we have to provide *limited* connectivity to those nodes. We also have to ensure that malicious applications hosted on the Marketplace will not to be reachable by any other mobile device on the Internet. Therefore, this scenario requires a segregated, controlled network.

4.3.4.2.2 Mobile App Marketplace Node

The node running the App Marketplace software that hosts mobile applications and distributes them to the end users who can access it through their smart devices.

4.3.4.2.3 Malicious Node

The malicious node accesses the Marketplace through an API and uploads malicious applications there that can infect mobile devices which use this platform.

4.3.4.2.4 Attack Traffic

In this scenario, the attack traffic is related with the downloading of a set of malicious applications by a Marketplace platform to the legitimate node. The traffic is generated by the legitimate node that browses the Markeplace and downloads mobile applications. Moreover another portion of traffic is related with the malicious applications that the malicious node (malware author) uploads to the Marketplace through an API provided by the Marketplace to developers.

4.3.4.2.5 Legitimate Node

The legitimate node is an Android device (e.g., smartphone or tablet), or an Android Emulator instance able to access the Markerplace platform that hosts and distributes the mobile applications. The node should be able to access the Mobile App Marketplace and download applications either through a client app developed for this purpose, i.e., for browsing and installing apps

(similar to the Google Play Store app), or by downloading applications directly from the server (e.g., via a web browser). Thus, the device/emulator should be preconfigured with the appropriate software.

4.3.4.2.6 Protection Mechanism The protection mechanism is a piece of software pre-installed in the legitimate node able to detect and generate notifications about malicious actions that an application may perform.

4.3.4.2.7 Metrics

In this section, we provide a description of the required metrics used for this scenario. Both *effectiveness* and *overhead* metrics are required. Below, we provide a more detailed description on how the metrics already discussed in Section 4.2 are being adapted to our scenario.

Effectiveness Metrics. The proposed defense mechanism targets a specific kind of malicious mobile application that can cause financial loss to mobile users, thus it is necessary to achieve high detection accuracy, otherwise users will suffer this financial loss. The system is based on labels that are built dynamically during the user experience with the system, so that any benign entity wrongly labeled as malicious (and vice versa) depends on the human factor. Another cause of this may be due to errors to the predefined lists of the system with the labeled entries (black/whitelists). Below, we provide a list of the adapted metrics (already discussed in Section 4.2) customized to work in our system:

Detection accuracy will allow the evaluation of the efficiency of our protection scheme in terms of number of malicious attempts detected over a set of real-world malicious applications (SMS fraudsters, premium-rate dialers). This metric is very important as it measures the degree of safety that the proposed protection mechanism can provide to the users.

Precision, recall will allow the evaluation of the efficiency of our system and its predefined structures, that is the memory protected whitelist and blacklist, over a mixture of labelled set of applications (i.e., benign and malicious).

Mitigation time measures the time it takes for our system to detect a malicious behavior of an application.

Overhead Metrics. The proposed system will detect any suspicious or malicious attempt of sending SMS or performing calls to premium numbers that was not initiated by the user of the mobile device. This poses an extra overhead to the OS of the device where the logic of our system is running, as the OS should intercept every SMS/call of the device. This overhead

should not be high in order for our system not to impact the normal user experience.

Detectors overhead assesses the extra computation power that our system introduces to the device in order to provide the necessary protection.

Average CPU load refers to the average CPU usage induced by our system to the device over a period of time. It is closely connected to the *detectors overhead* metric.

Memory consumption is related to the memory footprint that our system uses for keeping the whitelist and blacklist structures containing the premium numbers.

4.3.4.3 Sequence Diagram

In this section, we describe the scenario through a sequence of steps illustrated in Figure 4.4. Below, we describe each of these steps:

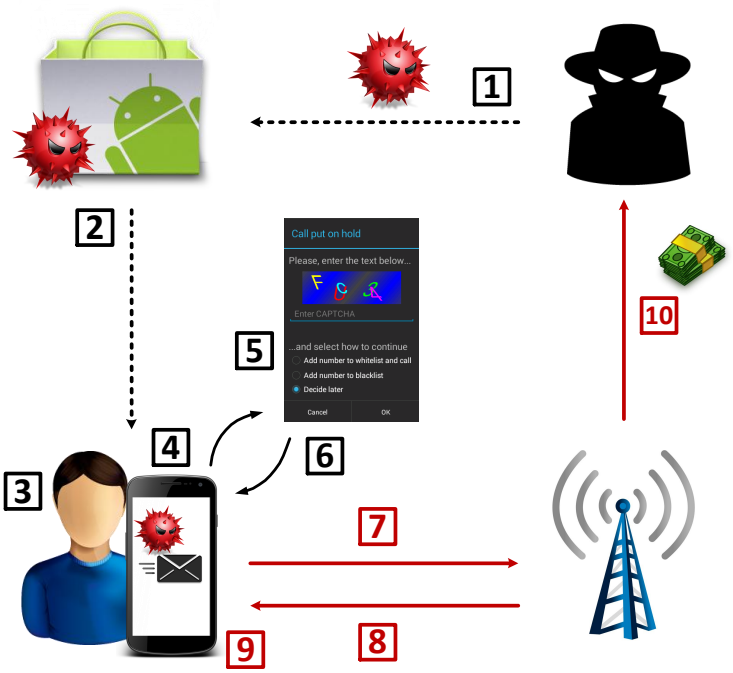

Figure 4.4: Scenario workflow for premium SMS/call fraud.

1. A malware author uploads a malicious application, that appears to be benign, to an Android marketplace.

2. The user visits this app marketplace, browse through the applications and downloads the one that the malware author previously uploaded there.
3. The user runs the application.
4. The application attempts to send a premium-rate SMS to a number unknown to the user.
5. The modified version of Android OS will detect this attempt and will pop up a dialog to the user, asking her to authorize or not the sending of the SMS and also to solve a CAPTCHA puzzle.
6. The user realizing that this number is unknown and this SMS was not initiated by her, refuses to send the SMS and adds this unknown number to the blacklist so as to prevent similar incidents in the future.
7. If the mobile device does not run the modified OS, the user will never get notified about the sending of the SMS to the malicious premium-number.
8. Another SMS will be received asking for payment confirmation which the user will not be able to see, as the malicious app will intercept it.
9. The malicious app will confirm the payment.
10. The malware author will earn money (stolen from the victim user).

5 Malware Campaign Mitigation

Malware propagation is currently a multilayer threat in which attacks against the infrastructure layer are a gateway to attacks against client applications and vice versa. This use case will validate the ability of solutions developed in the project to detect repeatable patterns showing infection of servers, resulting threat to clients and the threats to the infrastructure posed by client infections, identify the required mitigation steps, collect available information and react accordingly.

5.1 Description

A malicious campaign is a complex, multilayer and multistage threat, endangering a large portion of the global or regional network through coordinated, medium to long-term activity. The activities may involve many attack techniques, where the final goal of the campaign may remain hidden in the early phases.

Discovering and hindering a malicious campaign is a difficult goal involving several non-trivial steps. The individual actions have to be detected and recognized as malicious. Then the correlation between the actions has to be found in order to identify them as part of a larger phenomenon. Finally, the collected information must be analyzed in order to detect the weakest links in the scheme, allowing effective mitigation measures to be applied.

Countermeasures can be applied at different stages of a campaign. In most cases, concurrent deployment is advised, regardless of the current state of the campaign. The reason is that the different stages are usually not separated in time, but executed in a pipeline fashion. The initial stage is still in operation, seeking new victims, while the early captures are already used for next stages. The range of available countermeasures depends on

the details of a given campaign, especially the propagation and coordination mechanisms. They include actions such as blocking spam, server takedown, blackholing and others. In all cases, the most important aspect is correct identification of actionable information which can be used to block certain malicious activities: malicious URLs, keywords or other characteristics of spam, IP addresses or domain names used for C&C or as exploit servers, etc.

The use case is focused on showing such a complex analysis and its effectiveness in disrupting an ongoing campaign.

5.2 Metrics

5.2.1 Detection metrics

5.2.1.1 Detection precision

Type: Quantitative

Definition: The number of true positive detections out of total positive detections of malicious campaigns. In other words,

$(TruePositive)/(TruePositive + FalsePositive)$

Score: %

Measurement: See Section 6.6.3

5.2.1.2 Detection rate

Type: Quantitative

Definition: The number of positive detections based on data from a selected period, normalized to a single time unit (hours, days, etc. – to be determined in testing to provide the most expressive value).

Score: [n] (number of detections per period)

Measurement: See Section 6.6.3

5.2.1.3 Detection time

Type: Quantitative

Definition: Time between the start of analysis of the data and availability of actionable information.

Score: seconds/minutes/hours

Measurement: See Section 6.6.3

5.2.2 Mitigation metrics

5.2.2.1 Mitigation time

Type: Quantitative

Definition: Time between detection of a malicious campaign and mitigating the related activity.

Score: minutes/hours/days

Measurement: See Section 6.6.3

5.2.2.2 Actionable information count

Type: Quantitative

Definition: The metric measures the count of indicators generated by the analyses – the tokens usable for blocking the activity of a malicious campaign.

Score: [n] (the number of items)

Measurement: See Section 6.6.3

5.2.2.3 Mitigation effectiveness

Type: Quantitative

Definition: The metric measures the effectiveness of collected actionable information in blocking the activity of a malicious campaign by comparing the amount of traffic eliminated by providing the collected information to PEPs to the total amount of traffic recorded in the testing environment.

Score: %

Measurement: See Section 6.6.3

5.3 Scenario

5.3.1 Description

The scenario will demonstrate how the analyses implemented in NECOMA can be used to detect an ongoing malicious campaign, make the connection

between seemingly unrelated incidents, identify the activities related to the campaign and extract actionable knowledge from the available datasets, allowing effective reaction to the threat.

The scenario should include the following steps:

1. Detection of a possible campaign using the FP-growth approach (as described in deliverable D2.1) or other applicable methods. The datasets to be processed depend on the choice of methods for this step, e.g., in the case of FP-growth, the initial datasets would contain malicious URLs and domains.

2. Collecting related information from available datasets using the graph-based analysis. A campaign is considered to be likely if the collected information is rich enough and spans multiple datasets. Manual verification by an expert is advised in practice at this point, although automatic classification is possible.

3. Extraction of actionable information from data collected in previous steps (malicious URL templates, known exploit servers used in the campaign, more if available).

4. Depending on the capabilities and effectiveness of analysis methods available in the NECOMA platform, collection of further information, including e.g.,:

   - If a malware sample is identified, sandbox-based analysis of the sample.
   - Processing of the sandbox network traffic data in search of previously unknown C&C servers.
   - Identification of already infected machines based on backbone traffic data and/or using data collected after blackholing is applied.

5. Representation of the result of analyses in common interchange formats to be used either for filtering purposes or as identification of malicious nodes in the network to be eliminated.

6. Selection of campaigns for which actual malware samples are available for further testing; deployment of malware in the testing environment.

7. Application of collected actionable information in the testing environment.

   - May include further enrichment of data collected in the detection phase through the use of applicable analysis methods on data

collected within the testing environment to e.g., identify the infected machines. Additional actionable information may be used for mitigation as well.

- This is another point where in practice manual verification by an expert is advised, although – again – not strictly required. The risk of generating (potentially critical) false positives inherent in all automated systems makes expert knowledge invaluable, at least until a long period of operational use proves such oversight unnecessary.

8. Measuring the effectiveness of the applied filtering through verification of the remaining traffic after PEPs of the testing environment.

9. Eliminating the malware from the testing environment and additional tests with non-infected hosts to verify the level of false positives caused by the proposed mitigation measures.

The verification in the final steps is based on two-stage filtering. First, mitigation measures (such as filtering, sinkholing, etc.) are applied using the collected information as indicators of a campaign. Then, the remaining outbound traffic is measured and blocked to avoid impact on other systems. The remaining traffic, after eliminating known good connections (such as automatic updates) provides an estimate on the amount of malicious requests that were not effectively blocked.

5.3.2 Requirements of the Testing Environment

The main requirement of the first stage of the scenario (detection and analysis) is the availability of a snapshot of all datasets involved in the analysis, spanning a sufficient time period and holding enough data to make campaign detection possible.

Note that it is assumed that all analyses will in practice be performed on actual, current data in online datasets. The reason justifying the use of snapshots is the ability to perform manual analysis of the exact same data that were available to the analytical modules, making manual verification possible. An additional benefit of the approach is the ability to select a period with data interesting from the presentation point of view, showing successful application of as many analyses as possible against a single campaign.

Other than that, this stage does not require a separate testing environment apart from the deployed NECOMA platform. A separate deployment may be necessary if the main platform is already in operational use and cannot be redirected to work on a data snapshot for a short period.

The ability to perform a sandbox analysis of a sample requires some additional resources, either in the form of an additional machine to set up

a sandbox, or simply access to a preexisting, working sandbox environment capable of processing a few extra samples.

The verification in the mitigation stage requires several virtual machines forming a simulated network, with resources such as a DNS server with RPZ. The minimal setup should involve several infected virtual hosts and several non-infected hosts. Adequate virtualization environment must be available, including sufficiently capable hardware and network connectivity. All non-local traffic in this network must be routed through a single gateway to enable complete disconnection from external network when operating with active malware.

5.3.2.1 Metrics

The metrics collected during the test case are separated into two groups. The detection metrics characterize the detection process, showing the ability of NECOMA to detect an ongoing campaign. The mitigation metrics show the effectiveness of the application of results of the analysis performed in the NECOMA platform in hindering the campaign.

The main difficulty in the planned scenario is the large amount of data under consideration. Since the analysis is performed on a snapshot of actual NECOMA data, as opposed to a generated threat, there is no preexisting classification of data. The measurements involving values such as true/false positives/negatives have to be based on expert analysis. While it is assumed that the positive results may be verified in this way, it is not realistic to expect all the negative results to be reviewed, especially considering the limited resources committed. Therefore any metrics requiring good estimates of either true negative of false negative counts, while potentially applicable and informative, cannot be considered for this scenario. This is the reason why metrics such as detection accuracy or detection recall, quite general and measured in other scenarios, have been omitted.

Another important feature of this scenario is that the classification under consideration is not malicious vs. benign, but related to a campaign vs. isolated incident. Therefore the values considered are different than in the other scenarios – the focus is on the detection of coordinated campaigns and correct identification of connections between a given incident or data portion and a given campaign.

Also note that in case of time measurements, any delays introduced by the deployment of the collected malware samples in the testing environment have to be ignored, as they are specific to the test case and will not appear in the normal operation of the products of NECOMA – the reactions will be applied directly to the network.

The time-based metrics are not measured from the start of the campaign. There is an inherent delay between the start of the first activity of a campaign and the moment when the collected data contain enough related

incidents to make it possible to identify them as parts of a single campaign. This delay depends on the level of activity in the campaign – slow actions may delay the discovery of correlation even if all actions are correctly identified and available in the datasets. For this reason, the measurement of detection time marks the launch of the detection procedures as the starting point instead of the time of the first incident within the campaign (which may not be easy, or even possible to identify).

6 Demonstration Platforms

This chapter covers the demonstrators, i.e., testing platforms or environments that will host the evaluation of the resilience mechanisms developed in deliverable D3.4. According to the requirements described in the previous chapters, we present the specifications of the platforms that will accommodate the different proposed scenarios. Each platform will mention the hosted scenarios and will be described in terms of network, software and hardware specifications with respects to the underlying infrastructure and topology, as well as the different services or traffic to emulate. With respects to the evaluation aspect, the description of each platform includes the methods to measure the metrics proposed in earlier chapters.

6.1 High-speed Core Network Testing Platform

This section describes the demonstration platform for the *DDoS Mitigation as a Service* scenario, introduced in Section 2.3.3.

6.1.1 Infrastructure

In this scenario, a customer network is connected at the edge of an Internet Service Provider (ISP). This ISP network is also peered with content providers, as well as other peers where both legitimate users and attackers may lie. Some content providers will offer video streaming and HTTP services to the customer network, which constitutes the target of some DDoS attacks. As shown in Figure 6.1, the core router is placed at the core of our ISP network and is peered with content providers and other customer networks at its edges. Such peerings are represented by the links {`IBM1, IBM5`} and {`IBM2, IBM4`}.

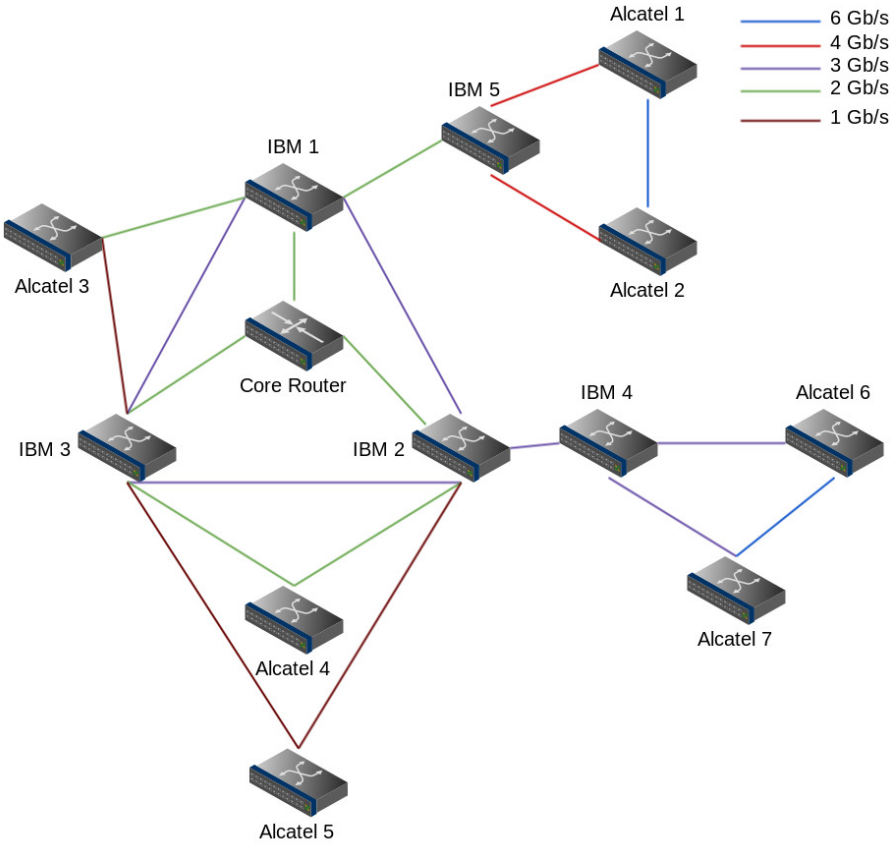

Figure 6.1: Topology of a High-speed core network platform hosting SDN-based DDoS Mitigation Scenarios

6.1.2 Topology

Switches lying in the ISP domain will be connected to an SDN controller, and the ones pertaining to the customer network domain will establish OpenFlow connections with another controller.

6.1.2.1 Hardware

The platform is composed of several equipments:

Router Cisco ASR 1002-X An MPLS-capable router with 6 Gigabit ports

IBM G8052 SDN-capable switches with 48 Gigabit ports

Alcatel OS6860 SDN-capable switches with 24 Gigabit ports

DELL Servers equipped with 8-core 2.9 Ghz CPU, 16 Go RAM and 10 Gb/s Ethernet NICs

6.1.2.2 Services

6.1.2.2.1 Web service An Apache server can be set up at one or the server or instantiated as a virtual service using cloud computing platforms such as OpenStack[1].

6.1.2.2.2 Video streaming A server can be dedicated to handle the video streaming flows. Since most recent services rely on HTTP, an Apache daemon serving video segments can be considered. Similarly, other media services can be considered and deployed on-demand via a cloud computing platform.

The network architecture, as well as the capacity of communication links between the routers are shown in Figure 6.1.

It features an internal ring of switches close to the core router that serves interconnection between peers. Links' bandwidth to provide an heterogeneous connectivity environment, in addition to the variety in switch vendors. The bandwidths provided by the links can be obtained through the aggregation of more or less physical links between the switches. This will allow for the diffentiation of paths in terms of quality of service or bandwidth, as proposed in the mitigation mechanism presented in Section 2.3.3. This way, we can virtually create bottlenecks at different strategic points, according to changes in the scenario.

6.1.3 Legitimate traffic generation

There are seven DELL servers available to generate both streaming and HTTP traffic. As per the hardware specifications of the servers, we intend to generate around 3 Gb/s of video streaming using a subset of the servers. There will be two machines running Apache servers and three others generating traffic for the clients. As for the HTTP traffic, we will use two machines to generate the HTTP requests. We have several tools to generate legitimate traffic,

TAPAS: Tool for rApid Prototyping of Adaptive Streaming algorithms This tool[2] simulates video streaming clients. Usually, one client will generate up to 1 MB/s of bandwith. Each DELL server will virtualize around 1000 clients and therefore reaching around 1 Gb/s.

Epload: Emulated page load This tool[3] has been specially conceived to run networking experiments. It allows to replay the loading of a same web page.

[1] http://www.openstack.org/
[2] https://github.com/ldecicco/tapas
[3] http://wprof.cs.washington.edu/spdy/tool/

6.1.4 Attack traffic generation

To generate attack traffic, we may use one or several of the following tools to simulate different kinds of DDoS attacks, with varying characteristics,

Hping3 Hping3 is an advanced ping tool used to generate quickly TCP/IP packets. It can be used for stress tests or DDoS attacks.

BoNeSi BotNet Simulator[4] allows a user to simulate a botnet on one machine and conduct DDoS attacks.

Tcpreplay Tcpreplay is not an attack tool but can replay pcap files. Therefore, you can replay an attack that has actually happened using its trace.

Considering the performance obtained by BoNeSi on a standard computer, we should be able to obtain around 1 million of packets per second on the network using our servers.

6.1.5 Measurement

The metrics considered in this scenario can be divided into three categories: network oriented, system oriented and user experience.

- A system metric is the one that can be measured using certain software such as the `top` command for CPU and memory usage.
- The network metrics can be measured by setting up a network probe. In particular, we will use the sFlow sampling facility available in the OpenFlow switchs. The performance metrics can be obtained through the statistical collection of incoming and outgoing network traffic.
- Using video streaming as legitimate traffic, we will measure the impact of the attack on the user experience. TAPAS provides for instance metrics such as the *time to rebuffer the video*. It is an important indicator as it measures for how long the video has been interrupted in order to fill the video buffer receiving the video segments.

6.2 MPLS-based DDoS Mitigation Platform

This platform is aimed to accommodate the scenario where DDoS defenses are pushed upstream, described in Sect. 2.3.1.

[4] https://code.google.com/p/bonesi/

6.2.1 Infrastructure

The platform consists of different components, described in the following subsections.

6.2.1.1 Network

- 2 * Cisco 7204 VXR routers
- 1 * Cisco 3660 router
- 1 * Cisco 2600 router

These routers are connected to each other by the means of one switch, allowing to change the topology by reassigning the router interfaces to different VLANs.

The routers are connected to each other with 1 Gbps links.

6.2.1.2 Service

The legitimate service is provided by a dedicated server hosting a web server and a DNS server.

6.2.1.3 Legitimate user

The legitimate user is represented by a single node (small desktop machine) that will be used to run tools allowing to measure the quality of the legitimate traffic.

6.2.1.4 Attacker

The attacker nodes on the platform are high-end servers equiped with Intel and Endace DAG network 10G network cards with several interfaces each. We can use several attacker nodes if required.

6.2.1.5 DDoS mitigation solution

The on-premise defense mechanism is represented by a 6cure Threat Protection solution, capable of filtering out DDoS attack traffic.

6.2.2 Topology

Figure 6.2 shows the current view of the platform with L3 connectivity. As physically, all router interfaces are connected to a switch, the L2 and L3 connectivity can be easily modified by changing the VLAN configurations.

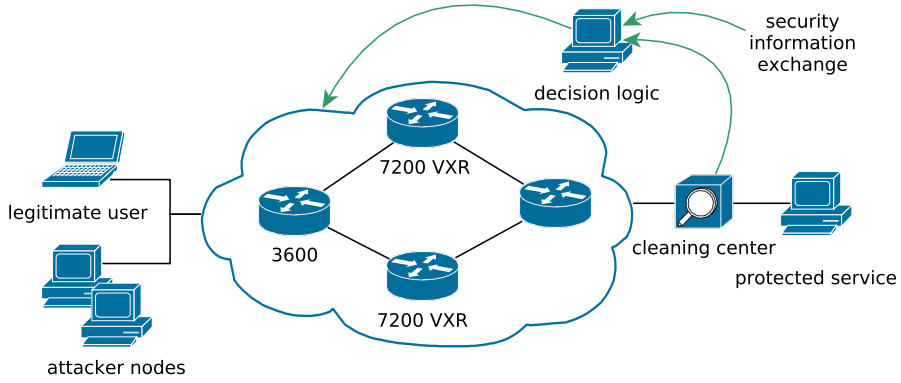

Figure 6.2: Current view of platform topology with the traffic generating nodes at left, the targeted service and on-premise protections on the left; the MPLS network in the middle with the currently available routers

6.2.3 Legitimate traffic generation

We use simple tools to generate legitimate traffic, such as

1. ping for basic latency measurements, and
2. Apache HTTP server benchmarking tool[5] for measuring application layer performance

We observe the latency and throughput characteristics before attack and then detect attack / defense mechanism impacts by observing variations in these characteristics during attack and mitigation phases.

6.2.4 Attack traffic generation

We are able to use different basic tools to generate attack traffic on the platform, such as

1. Hping3,
2. BoNeSi,
3. tcpreplay,
4. dagflood, a packet replay tool similar to tcpreplay for Endace DAG cards,[6]

[5] http://httpd.apache.org/docs/2.2/programs/ab.html

[6] http://www.emulex.com/products/network-visibility-products-and-services/endacedag-data-capture-cards

5. scapy,[7] a packet manipulation program.

For the type of attack traffic we need for our scenario (cf. Section 2.3.1), we can construct captures containing suitable attack traffic (e.g., handcrafting with scapy and/or capturing single requests and then modifying and multiplying them with scapy and/or tcprewrite) and replay it using tools like tcpreplay or dagflood. In other words, we will be replaying artificially generated traffic as it would be sent by the amplifier nodes towards the target - we do not have a functional set of amplifiers on the testbed.

6.2.5 Measurement

We will be able to collect packet and bit volumes on different interfaces on the routers, switches, and 6cure Threat Protection.

The tools used to generate the legitimate load allow measuring the packet and request latencies and packet losses/transaction failures, which allows also to characterize the denial of service effect of the attack as well as the mitigation effect.

We still need to design and/or configure the mechanism to allow measuring accuracy. This could consist of capturing packets at the mechanism egress point and comparing them to the capture used to generate the attack traffic (or to the capture made at the ingress point of the defense mechanism, in case of interactively generated attack traffic).

This approach for measuring accuracy would, however, face the following challenges:

Access to the traffic at mechanism egress and maybe ingress points could be implemented either at network interface level at the attacker / target nodes (with increased resource consumption), using port mirroring (with potential packet losses), or using network taps (requiring additional equipment).

Traffic capture and persistence to disk for post-processing and establishing the values for metrics could pose upper bounds for attack volumes based on disk throughput and storage capacity on the testbed; and would increase the risk of errors due to packet losses while capturing.

6.3 Cloud-based Testing Platform

This platform is a testbed using virtualization technologies. Since most of the components in the platform are built using software, it is not only a flexible environment for demonstrations, but also to provide a reproducible environment for future investigations.

[7] http://www.secdev.org/projects/scapy/

This platform accommodates several use case analyses described in Section 2.3.2, 3.3.1 and 3.3.2.

6.3.1 Infrastructure

The platform is mainly composed of virtual machines and virtual switches inside hypervisors. By using virtualization technologies and configuration tools, the platform is configured flexibly to meet various kinds of use case analyses.

6.3.1.1 Hardware

The platform consists of the following servers and network devices.

- 4 * DELL PowerEdge R320 servers
- 1 * DELL PowerEdge R310 server
- 2 * Cisco 3750 switch
- 1 * DELL S4810 switch

The four R320 servers are used as hypervisors and the R310 server is used as controller for the hypervisors. The hypervisors are connected to the 3750 switch via 1Gbps links and the controller is connected to the S4810 switch via 10Gbps link, and the 3750 and S4810 switch are connected via 1Gbps link.

6.3.1.2 Software

We designed the platform to configure and execute reproducible use case analyses. The following software are deployed for configuring environments, performing experiments, and measurements. All of the software are open source implementations.

- KVM : Kernel Virtual Machine[8]
  A hypervisor implementation on Linux kernel. It enables to run virtual machines on servers.

- libvirt : The virtualization API[9]
  A library to control KVM, network configuration, and storage by providing Application Programming Interfaces (APIs).

[8] http://www.linux-kvm.org
[9] http://libvirt.org/

- MAAS : Metal as a Service[10]
  A software for configuring bare-metal[11] servers automatically. In this platform, the software is used for installing Linux OS, Ubuntu 14.04, into hypervisor servers. When we add a new hypervisor server into the platform, all we need to do is running the software on a new bare-metal server. The software installs Ubuntu 14.04 OS into a new server and configures as a hypervisor.

- ANSIBLE : Simple IT automation[12]
  An automation tool to configure servers based on pre-defined configurations. In this platform, the tool is used for building experimental environments such as building VMs and configuring network topologies inside hypervisors.

- Cacti : The Complete RRDTool-based Graphing Solution[13]
  This software is used for measurements. It retrieves network status, usages of resources, and workloads by using SNMP.

We design and deploy the software into the platform to make it as automated and reproducible as possible for an experiment.

6.3.2 Building the Use Case Analysis Environment

6.3.2.1 Automated Testbed Installation

The platform is designed for executing various kinds of scenarios. The configurations should be installed as "playbooks", which are used for ANSIBLE. We can automatically install and configure the software of hypervisor, including guest VMs and networks described by playbooks. The format of playbook is YAML[14].

In case of DDoS mitigation described in 2.3.2, the topology of use case analysis in this platform is shown in Figure 6.3. In this figure, three networks are configured in hypervisors, for attackers, victims, and control of experiment. The hypervisors accommodate twelve VMs, one for attacker, one for victim, three for mitigation and control system, and seven for DNS servers. All of the VMs and network configuration are installed using ANSIBLE, so the configurations are provided as YAML files.

[10] http://maas.ubuntu.com/
[11] The terminology *Bare-metal* means a computer without an operating system in this context.
[12] http://www.ansible.com/
[13] http://www.cacti.net/
[14] YAML: Ain't Markup Language, http://yaml.org/

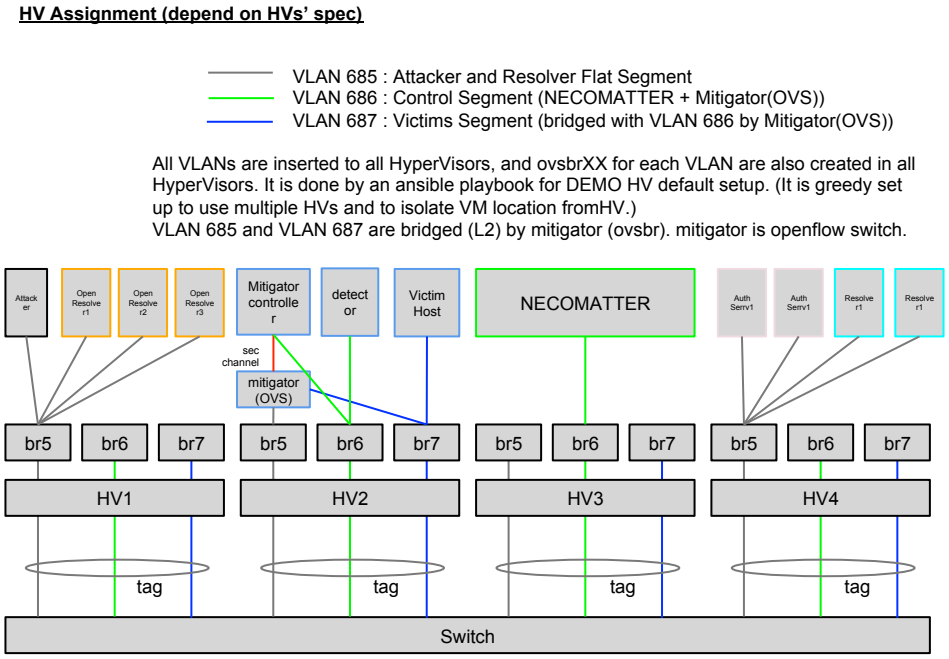

Figure 6.3: The network and VM topology for DDoS mitigation use case analysis.

The figure 6.4 presents an example of playbook, describing configuration explained above only for a node executing mitigation described by YAML. In this case, a couple of OpenvSwitch configuration will be deployed for the experiment with a few lines of code. Once we come up with such a concrete playbook, we can distribute this to the other environment so that other researchers can easily follow the instructions without knowing details of the original environment and conducted experiments.

6.3.2.2 Experiment Execution

The method of experiment execution depends on the each scenario. If the experiment can be executed in a non-interactive way, the scenario is also able to be written as an ANSIBLE playbook. Then the scenario is executed automatically, and the results are measured and captured. If the scenario needs some interaction in the scenario, it can be executed using any script language on the control server. In case of the DDoS mitigation use case, the scenario is written as shell scripts and executed step-by-step manually.

As mentioned before, this testbed platform is flexible for its configurations, and it is also flexible for executing experiments.

```
$ cat ansible-playbook/mitigator.yml
- hosts: mitigator
 sudo: yes

 tasks:
 - name: create mitigator openvswitch
   openvswitch_bridge: bridge=mitbr state=present
 - name: delete default ovsbrs
   openvswitch_bridge: bridge=ovsbr685 state=absent
 - name: delete default ovsbrs
   openvswitch_bridge: bridge=ovsbr687 state=absent
 - name: add vlan685 to mitbr
   openvswitch_port: bridge=mitbr port=vlan685 state=present
 - name: add vlan687 to mitbr
   openvswitch_port: bridge=mitbr port=vlan687 state=present
 - name: set openflow controller address
   openvswitch_controller: bridge=mitbr controller=tcp:192.168.3.14:6633
 - name: set sflow export
   openvswitch_sflow: bridge=mitbr collector=192.168.3.11:6343
 - name: set IP address of vlan686 for openflow secure channel
   shell: |
    ifconfig ovsbr686 192.168.3.40/24
```

Figure 6.4: An example of ANSIBLE playbook for DDoS mitigation scenario.

6.3.3 Measurement

The results of experiments can be measured and captured in this platform. Because the platform deploys virtualization technologies, network traffic can be monitored at almost any points, such as virtual network interfaces in VMs, virtual switches inside hypervisors, and physical network interfaces on hypervisors. Furthermore, the platform deploys cacti measurement tool, so the workloads and usages of resources in hypervisors are monitored automatically.

6.4 Smartphone Testing Environment

6.4.1 Infrastructure

In the scenario of SMS Fraud Protection, described in Section 4.3.4, a malware author uploads a malicious application to a mobile app marketplace and a user attempts to download this application using a smartphone or

tablet device (see topology diagram 4.4). The protection mechanism is running on user's device with Android OS version. The proposed protection can also run on an Android Emulator instance, instead of a device, so that a security analyst could be able to use it for performing analysis to suspicious applications.

6.4.2 Hardware

The hardware components that can be used in this scenario contain the means that a user may use to communicate with the mobile app marketplace, the servers that host the marketplace and the machine that the malware author will use to upload her malicious software to the marketplace:

- **Android Device:** A smartphone or table handheld Android device.
- **Servers:** A dedicated host running the app marketplace software.
- **Malicious host:** A machine to which the malware author has access in order to perform the uploading of the malicious apps to the markeplace.

6.4.3 User Devices

The devices that users use to connect to the mobile app marketplace should run a modified Android OS, version 4.4.3. So all devices that support this specific version of Android OS (version 4.4.3) can be used in the scenario.

6.4.4 Android Emulator Clients

Apart from native devices, Android Emulator instances can also be used in this scenario. Similarly to the native devices case, the Android Emulator instances should run the modified Android OS, version 4.4.3.

6.4.5 Mobile App Marketplace Server

The mobile app marketplace server in this scenario can be a copy of a real marketplace or a simplified form of it (e.g., a web server that contains a list of Android applications.)

6.4.6 Metrics Measurement

In this section we describe how we will measure the metrics described in the SMS fraud scenario 4.3.4.2.7.

6.4.6.1 Accuracy Metrics

Measurement methodology for metrics *detection accuracy, precision, recall*: To measure these metrics, a labeled dataset is required. That is, we need a set of malicious applications (that are confirmed to be malicious) and another one that is confirmed to contain benign applications. To measure the detection accuracy we have to install the applications of these two sets to a mobile device or an Android Emulator instance running our system and compute the metric values, that is $\frac{(TP+TN)}{(TP+TN+FP+FN)}$ for the detection accuracy, $\frac{TP}{(TP+FP)}$ for the detection precision and $\frac{TP}{(TP+FN)}$ for the detection recall.

6.4.6.2 Performance Metrics

Measurement methodology for metrics *mitigation-time*: To measure the above metrics, we need to add extra code to our system in order to keep logs of the events' (SMS/calls) timestamps, as well as log parsing techniques should be followed to find the actual time when an application attempted to send an SMS or to perform a call (through monitoring the logcat log). Moreover, extra modifications to other parts of the Android OS should included so as to be able to detect that times.

Measurement methodology for metrics *detection overhead, CPU load average, memory consumption*: To measure the *detection overhead* and the *CPU load average*, this we can CPU usage statistics provided by `proc` interface.

6.5 Tablet Testing Environment

6.5.1 Infrastructure

In the scenarios described in Section 4.3.1 and 4.3.2, a user connects to the Internet using a smartphone and/or a tablet device. The user visits legitimate websites and phishing ones hosted at Dell Servers. Defense mechanisms are provided to the smartphone systems.

6.5.2 Hardware

The platform is composed of several equipments:

Buffalo WZR-HP-AG300H An OpenWRT capable wireless AP.

Google Nexus 7 A smartphone/table powered by Android OS.

Windows Tablet A smartphone/table powered by Microsoft Windows 8.1.

DELL Servers 8 cores 2.9 Ghz, 16 Go RAM; 10 Gb/s Ethernet

EyeTribe An eye-tracking camera

6.5.3 Building Demonstration Environment

6.5.3.1 Services

6.5.3.1.1 Web service An Apache server will be configured to provide Web services at virtual computers on the servers.

6.5.3.2 Topology

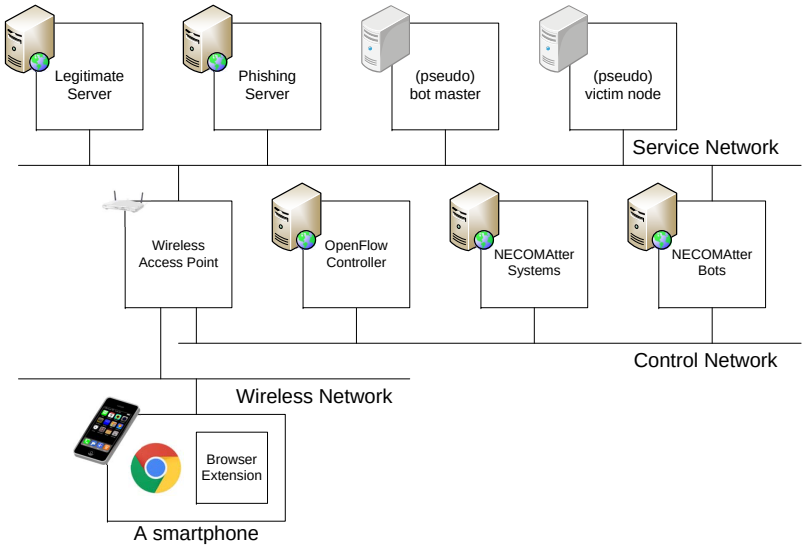

Figure 6.5: A topology of smartphone firewall

Figure 6.5 shows the topologies used in the scenarios described in Section 4.3.1 and 4.3.2. There are three networks as follows.

- *Service Network* allows to connect the nodes for service described in section 6.5.3.1. A node of NECOMAtter bots also connect to the network since several bot programs have to analyze cyber threats.
- *Control Network* is used for communication channel for the defense mechanism.
- *Wireless Network* is needed for connecting a smartphone device to the test environment.

6.5.3.3 Legitimate servers

In the phishing protection scenario, the legitimate websites used in the scenarios will be actual websites in the wild due to the difficulty of preparing EV-SSL certificates in the test environment.

In the smartphone firewall scenario, these servers are in the test environment for the Web services.

6.5.3.4 Phishing servers

In the phishing protection scenario, the phishing websites used in the scenarios will be a copy of phishing websites, and the server must be isolated from the Internet in order to avoid information leakage. Some phishing analysis modules will be required to store the check results of heuristics such as WHOIS results in the wild while for its analysis. The servers will run Apache Web server or our developed PhishCage system which focused on reproducing phishing website [9].

In the smartphone firewall scenario, it also stores malicious content.

6.5.3.5 (pseudo) bot master

This is used only for smartphone firewall scenario to prevent a smartphone to join DDoS. A pseudo bot program commands a smartphone in the host, so it is isolated to the Internet, in order to prevent sending backscatter packets.

6.5.3.6 (pseudo) victim node

This is also used only for smartphone firewall scenario to prevent DDoS. It is also quarantined to the Internet, in order to prevent sending backscatter packets.

6.5.3.7 Wireless Access Point

The wireless access point allows a smartphone device to connect to the service described in section 6.5.3.1. It also has to be capable to OpenFlow protocol by installing following tools.

OpenWrt: it is a Linux-based operating system for embedded devices such as wireless access points. [15]

Open vSwitch: it is a virtual switch that allows to OpenFlow controller with OpenFlow configuration protocol.[16]

[15] https://openwrt.org/
[16] http://openvswitch.org/

6.5.3.8 OpenFlow controller

This host is needed to install OpenFlow controller. We will select one of the following controllers.

POX: it is a python-based SDN controller to manage OpenFlow capable switches.[17]

RYU: it is also a controller for OpenFlow and an Operating System for SDN.[18]

6.5.3.9 Smartphone/Tablet devices

In the phishing protection scenario, smartphone/tablet devices will be used for participants. We will install Google Chrome and our developed extension in order to interact with an eye tracking camera.

In the smartphone firewall scenario, the devices try to join DDoS and/or browse malicious content such as phishing. The pseudo bot program will receive the commands from pseudo bot master and send DoS packet to the victim node. It also use Google Chrome as the Web browser.

6.5.4 Measurement

6.5.4.1 Phishing Protection

In this scenario, the effectiveness metrics are calculated by the questionnaires to the participants who browse the prepared websites in the test environment. It can be calculated by $\frac{(TP+TN)}{(TP+TN+FP+FN)}$, where TP denotes that a participant labels a phishing site as phishing, TN denotes that he/she labels a legitimate site as legitimate, FP denotes that he/she labels a legitimate site as phishing, and FN denotes that he/she labels a phishing site as legitimate.

The delay will be calculated at the browser extension, how long does a participant have to wait until he/she makes decisions. It can be calculated by $t_1 - t_0$, where t_0 denotes the time when the browser extension starts to initiate an eye-tracking camera, and t_1 denotes the time when the browser extension was committed the defense from the policy decision point implemented as a NECOMATTER bot.

The scenario-specific metrics are also measured by the questionnaires, as we described in Section 4.3.1.

[17] http://www.noxrepo.org/pox/about-pox/
[18] https://osrg.github.io/ryu/

6.5.4.2 Smartphone Firewall

For blocking malicious content, the delay will be measured at the wireless access point. It can be calculated $t_1 - t_0$, where t_0 denotes the time when the AP requests an OpenFlow controller to process the packets from Web browser in the device, and t_1 denotes the time when the controller replies the response.

For DDoS prevention, the time to recover will be also observed at the wireless access point. It can be calculated $t_1 - t_0$, where t_0 denotes the time when the bot master tries to join a smartphone to DDoS campaign, and t_1 denotes the time when the campaign was mitigated by the AP.

The scenario-specific metrics are also measured by the questionnaires, as we described in Section 4.3.2.

6.6 Campaign Testing Environment

6.6.1 Infrastructure

In the malware campaign mitigation scenario, described in Chapter 5, the malware collected in the detection phase of the scenario is deployed in a controlled environment to observe the ability of NECOMA to block its activity.

6.6.2 Hardware

The hardware used in this scenario needs to be able to provide the required number of virtual machines, some of which are infected, some clean and a few may play special roles, e.g., the DNS server.

NASK, as the unit responsible for execution of this scenario, has sufficient hardware ready for deployment, namely:

- One dedicated physical machine (2x Xeon processors, 4 cores each),
- 10-20 (as needed) virtual machines within existing infrastructure of CERT Polska,
- Network hardware providing connectivity for the machines.

The available hardware is sufficient to run 10-30 virtual machines forming the testing environment without performance issues – more than enough, considering the requirements of the scenario.

The proposed network infrastructure, as seen from the individual virtual machines, is a single network segment, with a selected machine as the gateway, providing the necessary filtering.

6.6.3 Measurement

Most of the metrics for this test case can be measured directly during the test, some however require expert analysis.

Detection metrics do not require the testing environment to be measured, as they are concerned with analysis of real data. The detection rate and time can be measured directly, with the special consideration that for the detection time to be correctly established the analysis must either run on demand (offline) or – in the online, constant monitoring mode of operation – be paused before starting the test to obtain a well-defined timestamp for the beginning of processing.

The detection precision metric is calculated from results of expert analysis of detections after the test. In case of a large number of detections the metric may be approximated from a sample limited by resources available for verification.

Mitigation metrics include the results obtained from the testing environment. The mitigation time is a simple sum of the detection time and the directly measurable time of deployment of the discovered information to PEPs. The time required to deploy the malware to the testing environment is ignored on purpose, as it is not part of the envisaged normal operation of the platform.

The actionable information count is also directly measurable as the number of rules actually deployed in the testing environment for a given campaign.

The mitigation effectiveness is measured in the testing environment by counting the actions performed by PEPs (in terms of blocked requests or connections) and the requests / connections that were allowed to propagate.

7 Conclusion

The large number of planned demonstrators and the wide scope they cover highlight the ambition of the *NECOMA* project. The platforms presented in this deliverable will allow the assessment of the performance and usability of the resilience mechanisms developed in workpackage 3. In addition to testbed evaluation, we will, in some scenarios, validate the information pipeline from dataset collection to PEP reconfiguration, with a possible feedback to the collection probes. In particular, the "Malware Campaign Mitigation" scenario aims at demonstrating the whole workflow using datasets contributed by other partners. Other demonstrators may demonstrate whole or part of the workflow, with focus on the protection of designated assets.

Indeed, no scenario is complete without the inclusion of a specific asset to protect, towards which the attack is direct, and on which the impact of the countermeasure will be measured. As our ambition is to protect not only the network infrastructure, but also the endpoint, including the user, our demonstrators span a consistent number of use cases, further refined into several scenarios, in order to represent diverse settings of attacks, technologies and assets. Taking into account the asset to protect, we have proposed or defined a number of metrics to assess the performance and usability of our resilience mechanisms. While some metrics are common across all scenarios, and even across use cases (*accuracy*), other metrics allow us to get more insights on particular scenarios, with respect to the assets to protect, or the particular technology used to carry out the reconfiguration or response.

Finally, the goal of each scenario may vary from one use case to another: while "DDoS Mitigation" clearly states its objectives, "Botnet Introspection" attempts at the extraction of information that will be later used by PEPs to combat botnets or simply detecting botnet activity. Despite such discrepancies, we have proposed to mutualize the demonstrators for some use cases that can be emulated on top of the same testbeds. Similarly, some software specifications, e.g., to generate legitimate or malicious traffic, have been

the subject of discussions among partners, and shared to be used in several different use cases.

While the details of some of the use cases are still vague and will be finalized in next deliverable D4.2, the requirements and assumptions are sufficiently detailed to drive the specifications of the platforms documented in this deliverable. Indeed, the attacks, assets and PEPs have often been surveyed in previous deliverables, leading to much maturity in the designed use cases. Remaining details are part of the implementation and performance of our resilience mechanisms, that are still ongoing. Therefore, we reserve the right to amend the scenarios and their number in subsequent deliverables.

Bibliography

[1] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, 2003.

[2] Akamai. Prolexic Quarterly Global DDoS Attack Report Q1 2014. Technical report, Prolexic, 2014.

[3] A. Belenky and N. Ansari. On Deterministic Packet Marking. *Comput. Netw.*, 51(10):2677–2700, July 2007.

[4] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2013–2018, June 2014. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf.

[5] R. Hansen, J. Kinsella, and H. Gonzalez. Slowloris HTTP DoS, 2009.

[6] G. N. Koutepas, F. Stamatelopoulos, and V. Maglaris. Distributed Management Architecture for Cooperative Detection and Reaction to DDoS Attacks. *Journal of Network and Systems Management*, 12:73–94, 2004.

[7] Lookout. State of mobile security, 2012. report: https://www.lookout.com/static/ee_images/lookout-state-of-mobile-security-2012.pdf.

[8] D. Miyamoto, T. Iimura, G. Blanc, H. Tazaki, and Y. Kadobayashi. EyeBit: Eye-Tracking Approach for Enforcing Phishing Prevention Habits. In *Proceedings of the 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, Sept. 2014.

[9] D. Miyamoto, Y. Taenaka, T. Miyachi, and H. Hazeyama. PhishCage: Reproduction of Fraudulent Websites in the Emulated Internet. In *Proceedings of the 1st Workshop on Emulation Tools, Methodology and Techniques (EMUTools)*, Mar. 2013.

[10] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[11] T. Peng, C. Leckie, and K. Ramamohanarao. Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems. *ACM Comput. Surv.*, 39(1), Apr. 2007.

[12] RSA Online Fraud Resource Center. The Current State of Cybercrime 2014. Available at: http://www.emc.com/collateral/white-paper/rsa-cybercrime-report-0414.pdf, 2014.

[13] R. Sahay, G. Blanc, Z. Zhang, and H. Debar. Towards autonomic ddos mitigation using software defined networking. In *Proceedings of the NDSS Workshop on Security of Emerging Network Technologies (SENT15)*, 2015.

[14] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. *SIGCOMM Comput. Commun. Rev.*, 30(4):295–306, Aug. 2000.

[15] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *Proceedings of the Network and Distributed System Security Symposium*, Feb. 2011.

[16] A.-D. Schmidt, H.-G. Schmidt, L. Batyuk, J. H. Clausen, S. A. Camtepe, S. Albayrak, and C. Yildizli. Smartphone malware evolution revisited: Android next target? In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pages 1–7, 2009.

[17] D. Sterne, K. Djahandari, B. Wilson, B. Babson, D. Schnackenberg, H. Holliday, and T. Reid. Autonomic Response to Distributed Denial of Service Attacks. In *Recent Advances in Intrusion Detection*, volume 2212 of *LNCS*. Springer Berlin Heidelberg, 2001.

[18] V. Svajcer. Sophos Mobile Security Threat Report. Available at: http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-mobile-security-threat-report.pdf, 2014.

[19] M. Yu, Y. Zhang, J. Mirkovic, and A. Alwabel. SENSS: Software Defined Security Service. In *Presented as part of the Open Networking Summit 2014 (ONS 2014)*, Santa Clara, CA, 2014. USENIX.

[20] S. T. Zargar, J. Joshi, and D. Tipper. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys and Tutorials*, 15(4):2046–2069, 2013.