Resilience and Survivability for future networking: framework, mechanisms, and experimental evaluation

Deliverable number	2.4a
Deliverable name	First draft of the learning framework for resilient networks
WP number	2
Delivery date	31/10/2010
Date of Preparation	25/2/2011
Editor	M. Schöller (NEC)
Contributor(s)	M. Schöller (NEC), T. Taleb (NEC), J. Lessmann (NEC), S. Martin(ULg), A. Fischer (UP), H. De Meer (UP)
Internal reviewer	P. Smith (ULanc), B. Plattner (ETH)

Summary

This deliverable presents the ResumeNet project's work objectives on the evolution framework, which implements the background control loop of our $D^2R^2 + DR$ strategy. It is based on the ResumeNet Deliverables D2.1b, D2.2b, and 2.3b, which presented the real-time control loop part of our strategy. The work items on the evolution of a networked system use information from past cycles of this real-time control loop to learn for improving future adaptation, and to extract new features from the collected data, such as placement strategies of multipath trunks. Note, that we consider learning schemes as an essential building block of this system evolution, but do not envision to propose new learning algorithms. In this deliverable we focus on the whole background control loop enabling evolution instead of learning only.

Contents

1 Introduction

The background control loop of our general D^2R^2+DR strategy (see Figure 1) aims at improving the real-time control loop by evolving the system based on learning from past adaptation cycles. This evolution of the system can be autonomic or can require human interaction. An example of an autonomic evolution is a self-improving algorithm, which changes configuration settings for future deployments, as we envision for the rope-ladder routing [LSZB10] scheme. Human interaction is required if the impact of challenges have been underestimated during the risk assessment process [SS09], and thus new defensive measures, challenge detection mechanisms, or new adaptation strategies have to be introduced to the system.

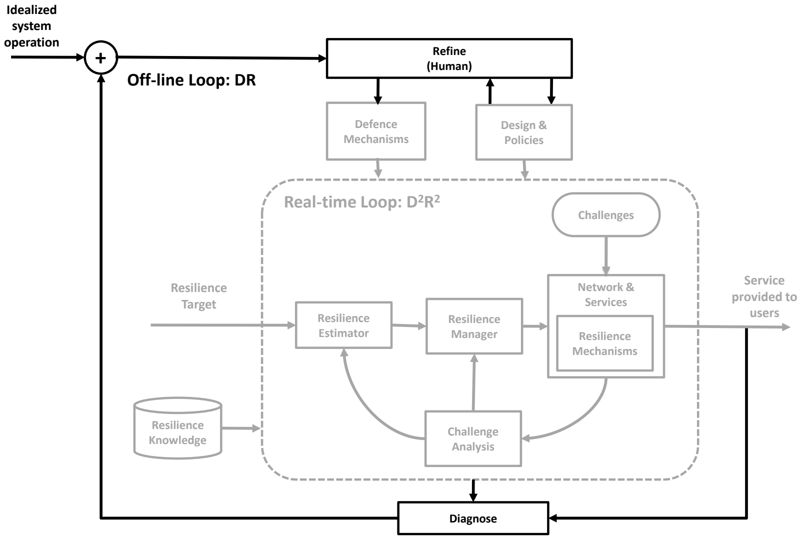


Figure 1: The ResumeNet control loops, with the evolution loop highlighted.

Independent of the actual refinement method applied, information and features of past operations has to be accessible. One way of storing the information required for this learning stage can be provided by the DISco system [MSF10], which has been designed for this purpose. DISco can store

- measurement data collected throughout the networked system;
- alarm information from challenge identification engines, which have correlated the measurement data to identify ongoing challenges;
- remedy descriptions, which characterise the adaptations applied in response to the challenge;
- context information form arbitrary information sources, which have impacted the remedy selection process; and
- remedy success reports, detailing the success of the applied remedy to deal with the challenge in the given context.

As we have pointed out in [DSS$^+$10], the full set of features provided by the challenge detection and remediation framework cannot be implemented for every scenario. Thus, the learning schemes developed in this activity must be able to deal with different sets of information provided or have to be specialized for certain deployments.

We use a bottom-up approach to identify necessary components of the background control loop. Thus, we start by investigating five different resilience mechanisms from different domains to develop diagnosis and refinement features, leading to an evolution of these mechanisms. The set of mechanisms is wide-ranging, spanning all stages of the ResumeNet real-time control loop: defensive measures, challenge detection mechanisms, and challenge remedies. In addition, we aim to gain a more systematic understanding of the requirements on the information to be made available to and the features which have to be provided to enable system evolution. The results from these activities will be presented in an updated version of this deliverable at the end of the project.

The remainder of this deliverable presents the five resilience mechanisms and how refinement based on learning could be used to evolve them. The mechanisms are:

1. SPam over Internet Telephony (SPIT) Prevention;
2. DTRAB (Detection and TRAcing Back of attacks on encrypted protocols);
3. the QoS^2 framework – a framework to integrate Quality of Service with Quality of Security;
4. the Rope-Ladder Multi-path Forwarding Protection Scheme; and
5. Virtual Service Migration.

2 The use of DISco for Diagnosis and Refinement

In deliverable D2.2 [MSF10], we have introduced the Distributed Information Store for Challenge and their Outcome (DISco), and presented how it can be used in the real-time control loop to relay events and provide context information required by detection engines and policy-driven remedy selection. Simultaneously, we want DISco to capture relevant information about events, their interdependences and their context to ease the tasks of diagnosis and refinement. Through the experiments described in this section, our aim is to investigate the limits of automated refinement as well as the implication of DISco aggregation on machine learning (ML) powered detection and identification of challenges.

A first task that can be automated in diagnosis is the assessment of the remediation action that has been taken. According to our challenge detection and remediation framework, components that detected the onset of a challenge published a *start-of-challenge* report in DISco, and the activation of a remediation mechanism is equally advertised through DISco *start-of-remediation* and *end-of-challenge* reports. Using these notifications as time marks, the diagnosis process will compare the evolution of target metrics (e.g. transaction serviced over time) and decide whether the high-level objectives (e.g. 99.99% availability) have been met. Mismatches of this objective may need to be reported to a human operator, who should be able to retrieve a detailed report of what automated decisions were taken, what were the characteristics of incriminated traffic, and proceed with manual refinement of the design. Whenever felt advisable, DISco should also support fully-automated refinement.

The two following scenarios illustrate the requirements for diagnosis tasks where human interaction is required.

- The existing policies did not yield to the desired behaviour. The automated diagnostic process must then ensure that the activated policies can be identified, and that it collected the context information required to evaluate a posteriori why other policies where disregarded or inhibited.

- The faced challenge was a new variant of an attack designed to evade existing detection mechanism (usually by altering its signature and possibly also changing the attack traffic to look more like normal traffic). The system must be able to store the suspicious flows' properties at least until a more informed actor can actually flag the flow as being malicious.

As we will detail below, there are situations in which end-users, although not experts, can help labelling the actions of a fully automated system. This allows reinforcement learning to be applied, enabling the detection and remediation tasks to evolve. Our distributed store should assist this by linking reported false positive/negative to their corresponding feature vector and enable exploration of additional features that could help discriminate unwanted traffic from legitimate traffic.

2.1 The SPam over Internet Telephony use case

In VoIP terminology, *SPIT* is the equivalent of e-mail SPAM: an unsolicited call that, when answered, delivers nothing but a pre-recorded advertisement. However, unlike in SPAM handling, decisions of whether a call should be forwarded or blocked need to be done without knowing the content of the message, before we let the target device ring.

In [Nas09], the feasibility of machine learning-based detection of SPIT has been shown, and an initial list of features useful for such detection is discussed, that may serve as basis for our experiments. Two techniques have been evaluated – namely Naïve Bayesian Networks and Support Vector Machines – that can provide onset (SPIT) challenge detection based on SIP statistics.

An interesting point is that when the confidence level of the classification is too low, it is possible to apply "soft" mitigation techniques – mainly simple Turing tests or vocal CAPTCHAs – that test a call before forwarding it to its recipient. Every passed test indicates a potential false positive. Similarly, an appropriate feedback mechanism – a "red button" for flagging calls – on end-user devices can be used to report and log false negatives (that is, SPIT that made it through the various defence and detection mechanisms). These two unsupervised reinforcements provide the required information to automatically collect and analyse occurrence of detector misbehaviours.

Our aim with the SPIT example is to illustrate how reinforcement can be applied to adapt to evolution of challenges. Here, we assume that this evolution is a reaction of the spammer who modifies his strategy to circumvent a deployed defence or remediation mechanism. A protection that requires the caller to press one of the dialing buttons on his phone, for instance, could be passed through with a probability of 10% if the spammer randomly dials one digit prior to talking. The effect of this modification is an increase of false negative (user-reported unsolicited calls) reported in DISco.

2.2 Exploiting DISco content for Machine Learning

In a simple potential approach, every incoming SIP call is classified into regular/SPIT and subsequently either forwarded to a phone or rejected. The classification uses an input vector that includes some features directly extracted from the request, grouped with locally collected statistics. Honey pots and in-lab attack botnets can be used to train the classification system – for instance, a support vector machine. The presence of DISco, and of the overall resilience architecture, allow us to capture recent history or to schedule the capture of additional information on demand. This potentially allows to identify new (and possibly more discriminating) features through on-line exploration.

The locally collected statistics (such as average call duration or number of incoming calls per second) can also be completed by subscribing to similar observation of peer/neighbour gateways. It is also possible to use the DISco `lookup` feature to retrieve recent history that could not be anticipated to be useful (e.g., transport session establishment details as we observe that the request carried over that session is suspicious). An automated correlation test of the additional information against the corrected classification output has the potential to refine the detection systems and the remediation decisions.

We argue that the ML classification approach might be insufficient when attackers actively search for a way to bypass deployed defence and detection measures. The argument is a parallel to the observation that identifying bots amongst regular WWW users using browsers signature can only work temporarily. Newly introduced (regular) browsing software will change the signature and the set of available features for identification. Similarly, development of new attack scripts better mimic behaviour of regular software with the objective of evading detection. The argument can easily be translated to the case of SIP phones and SPIT detection evasion strategies, suggesting that on-line reinforcement learning would be a better option (although more experimental) than off-line refreshing of the training set using additionally collected samples.

Seeing the SPIT mitigation problem as a pure classification problem assumes that we can provide a false positive rate that is low enough to indeed drop calls that are classified as SPIT. While accurate detection of SPIT calls based solely on observable features available when the call request appears may not be possible, it is interesting to note that in this context, a large panel of actions are actually possible, from "please wait" messages to small puzzles, that simultaneously reduce the probability that a real call gets dropped and that a SPIT call made it through the system to an end-user. Moreover, many of these elementary actions can be chained, and new variations can easily be deployed in replacement or in addition of existing ones. As the strategy of spammers is fundamentally unknown, the role of the SPIT protection system becomes to select the "filter" (or sequence of filters) through which a call will go in order to reduce the likelihood that a (believed) SPIT call makes it through, while reducing the annoyance (time-to-talk) for regular calls. Under that perspective, the problem is closer to a *multi-armed bandit* problem [JEMP09]:

- the control system is given a large range of possible actions (levers, for instance a sequence of safeguard measures before the call is delivered to an end-user);
- the control system able to observe the environment (the input vector mentioned earlier);
- the effect of the decision in a given context can be assessed, and a reward/punishment is returned to the control system as feedback. The correct answer(the lever that would have returned the highest reward in that situation), however, is not given.
- the system is coded to maximise the sum of rewards received over time, reapplying decisions that led to good rewards when a situation close to a known one is encountered, but keeping the flexibility to explore levers that haven't been tried yet.

The exact formalisation is still to be done.

2.3 Other Things of Interest

The SPIT detection technique presented in [Nas09] essentially inspects events that happened over the last k minutes and raise an alarm if there is evidence of SPIT (or another anomaly) in this time window. Information gathered is complete and analysis starts only when it has been fully retrieved on the system running ML classification. Using DISco to relay events is a first step towards a real-time handling of call requests. However, the limits of such an approach and the requirement of machine-learning algorithms on the availability and freshness of the information need additional study. The information published in DISco might have been discarded, aggregated or simply has an access time over the affordable threshold. What implication such probabilistic information retrieval has on the reliability of the classification (or more generally risk of forwarding the call) should be investigated. Another interesting interaction is to determine to what extent the machine learning output can guide the lifetime management strategies of DISco.

2.4 Work plan

The MADYNES team of INRIA Nancy has recently developed and published a collection of open-source tools for monitoring and defending a SIP network[1], plus tools that mimic the behaviour of a VoIP botnet in lab[2]. Combined with existing open-source VoIP systems such as the Asterisk PBX and the OpenSER proxy server, they allow one to build the appropriate environment for testing various VoIP attacks and the corresponding defences using commodity hardware only.

The adaptation of the core algorithms of DISco to a testbed environment – as opposed to the OMNet++ simulator module – is already in progress, and will be followed by the alteration of the VoIP software mentioned above so that reports get published into DISco.

The pre-existing datasets on SPIT classification will be completed by measurement from a local deployment of the tools, and will serve the purpose of illustrating the evolution of detection and remediation mechanisms. We will augment the VoIP testbed to include false negative and false positive reports in SPIT classification from the user agents (human assisted *diagnosis*), and illustrate the collection of new training data and automatic identification of features that best discriminate SPIT calls (*refinement*). This setup should be ideal to investigate the potential benefit of a reinforcement learning-based approach to management of anti-SPIT measures. These experiments will also provide the required ground to further study adaptive lifetime management strategies and the implication on the machine learning algorithm of partial information availability.

Additional resources would allow more challenges to be studied (e.g. SIP Flooding attacks, SIP/DNS interaction) and extension to other kind of volume-significant anomalies (including VoIP call floods, DDoS, flash crowds).

3 Self-Learning features in DTRAB

The unbridled growth of the Internet and network-based applications has contributed to enormous security leaks. Even the cryptographic protocols, which are used to provide secure

[1] http://gforge.inria.fr/projects/secsip

[2] http://gforge.inria.fr/projects/voipbot/

communication, are often targeted by diverse attacks. Intrusion Detection Systems (IDSs) are often employed to monitor network traffic and host activities that may lead to unauthorised accesses and attacks against vulnerable services. Most of the conventional misuse-based and anomaly-based IDSs are ineffective against attacks targeted at encrypted protocols, such as SSL or HTTPS, since they heavily rely on inspecting the payload contents. To combat against attacks on encrypted protocols, we proposed an anomaly-based detection system that uses strategically distributed Monitoring Stubs (MSs). The MSs, by sniffing encrypted traffic, extract features for detecting attacks and construct normal usage behaviour profiles during a learning phase. Upon detecting suspicious activities due to deviations from these normal profiles, the MSs notify the victim servers, which may then take necessary actions. In addition to detecting attacks, the MSs can also trace back the originating network of the attack. So the MSs can detect and trace back attacks against cryptographic protocols, there are four modes of operation: learning, detection, alert, and traceback phases.

In this activity, we will focus mainly on the learning phase of the DTRAB mechanism (Detection and TRAcing Back of attacks on encrypted protocols), without mentioning previous work done in the area. A thorough discussion on how machine learning is used in the area of network security, particularly detection and trace back of attacks on encrypted protocols, can be found in [Fad10].

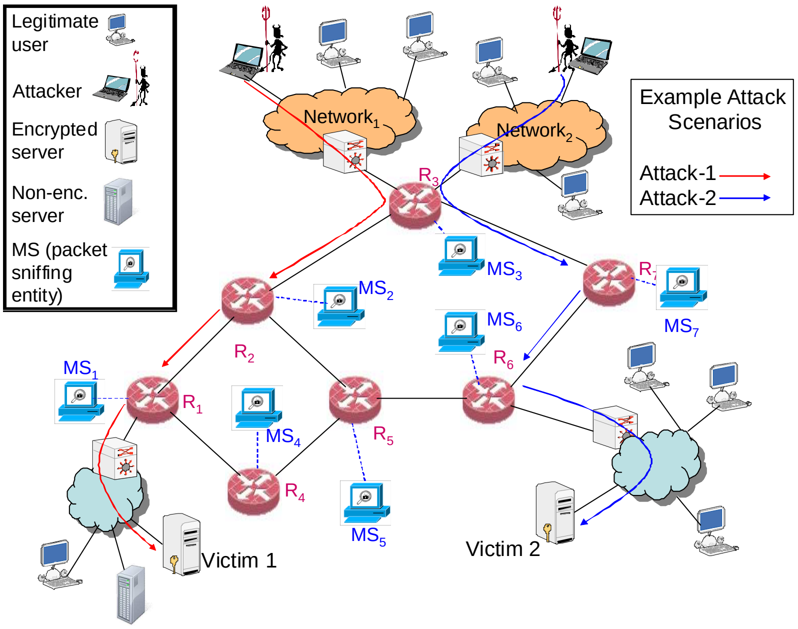
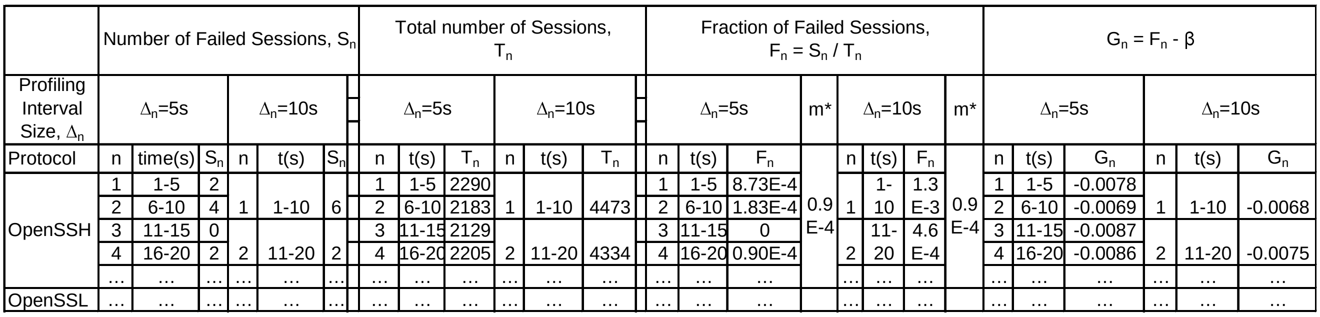


Figure 2: A Sample architecture with added MSs in DTRAB.

An example scenario of the envisioned network topology in DTRAB is shown in Fig. 2, which consists of a number of servers running services based on both encrypted and unencrypted application level protocols. Users from an untrusted network or from the Internet may connect to any one of these servers. Seven MSs are placed aside the network elements. The MSs, by sniffing, monitor the traffic headers, but do not inspect the payloads. When an attack is launched by a host (in Network-1), say from the untrusted network to victim server 1, MS_3, MS_2, and MS_1 consequently observe an influx in abnormal protocol operations interpreted as an attack-feature. In the remainder of this section, we describe how based on self-learning, the MSs effectively detect attacks against encrypted protocols and try to trace back the attacker. Furthermore, by specifying the normal operational modes and Request for Comments (RFCs)



	Number of Failed Sessions, S_n						Total number of Sessions, T_n						Fraction of Failed Sessions, $F_n = S_n / T_n$								$G_n = F_n - \beta$					
Profiling Interval Size, Δ_n	Δ_n=5s			Δ_n=10s			Δ_n=5s			Δ_n=10s			Δ_n=5s			m*	Δ_n=10s			m*	Δ_n=5s			Δ_n=10s		
Protocol	n	time(s)	S_n	n	t(s)	S_n	n	t(s)	T_n	n	t(s)	T_n	n	t(s)	F_n		n	t(s)	F_n		n	t(s)	G_n	n	t(s)	G_n
OpenSSH	1	1-5	2	1	1-10	6	1	1-5	2290	1	1-10	4473	1	1-5	8.73E-4	0.9E-4	1	1-10	1.3E-3	0.9E-4	1	1-5	-0.0078	1	1-10	-0.0068
	2	6-10	4				2	6-10	2183				2	6-10	1.83E-4						2	6-10	-0.0069			
	3	11-15	0	2	11-20	2	3	11-15	2129	2	11-20	4334	3	11-15	0		2	11-20	4.6E-4		3	11-15	-0.0087	2	11-20	-0.0075
	4	16-20	2				4	16-20	2205				4	16-20	0.90E-4						4	16-20	-0.0086			
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...		...	...	...		...	...	...	...	...	...
OpenSSL	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...		...	...	...		...	...	...	...	...	...

* m = ($\sum F_n$)/(number of observations during n)

Figure 3: A part of the database format created at a Monitoring stub (of DTRAB) for profiling.

specifications of different protocols in the MS' databases, this approach may also be extended to detect attacks against standard application-level protocols. For ease of understanding, we include below two examples demonstrating the deviations from normal operations of two encrypted protocols (SSH and HTTPS, respectively) under attack that may be considered as possible attack features. These selected features are extracted from the RFCs manually and are fed into the MS' databases.

Since an MS is only a packet-sniffing entity located alongside a router, it does not slow down the network traffic. In case of application level protocols, it is a trivial task to sniff both the network packet headers and the payload-contents, and inspect and analyse the information afterwards. For encrypted protocols, an MS needs to adopt a different approach. An MS uses the TCPDUMP tool to monitor the TCP headers which are not encrypted. For example, in order to detect a failed SSH-session due to a password-based attack against SSH-based services on port 22, an MS requires to know how the SSH protocol works at the transport layer. At first, a client attempts to establish a connection to the server by sending a SYN packet. The server acknowledges this by sending an ACK and a SYN packet of its own. If the client manages to successfully log onto the server and wants to quit, the client will initiate the FIN packet first. This is a normal mode of operation in SSH. On the contrary, if the server initiates the FIN packet first, it indicates that the server is shutting down the connection because of either an invalid attempt to access the service or a time-out. An MS monitors such connection flows and when it discovers that the server is the originator of the FIN packet soon after the connection attempt, it recognises a deviation in the protocol's normal mode of operation and deems that event as a "failed session".

By analysing encrypted web traffic flows using SSL/TLS, an MS may also see abnormal protocol operations in terms of the ratio of the request size to the corresponding response size. In case of "low interactive attacks", which are mostly launched against webservers offering HTTPS services, an MS may look to extract such information and consider it as a potential attack-feature. With non-encrypted HTTP traffic, this can be readily computed by inspecting the packet contents. However, extracting the request and response sizes from the HTTPS headers in encrypted traffic is indeed difficult, since these headers are also encrypted. Furthermore, random paddings of up to 255 bytes are added to the packets in SSL/TLS.

For this purpose, the MSs sniff packets destined for port 443 and look for client requests. If packets are observed continuously, they are considered to belong to a single activity, such as clicking a URL, or requesting a file to download. As the MS determines such an activity, it starts reconstructing the corresponding TCP sessions from the headers. Once the TCP sessions are established, the MS can decode the SSL/TLS session to obtain the sizes of the request and corresponding response from the server. This is possible because every SSL/TLS packet is structured in such a manner that the "Fragment" header of the "Record Layer" is encrypted while the rest three headers, namely "Length", "Version", and "Type" are not encrypted. The "Length" header provides an estimate of the size of either the request or response packets, depending on the directionality of the connection in the considered flow.

It should be noted that each MS maintains a database of RFCs specifications pertaining to the usage of various encrypted and non-encrypted protocols. The functionality of an MS is manifold, including learning normal profiles, monitoring for deviations from normal protocols operations, generating alerts, and finally tracing back the attacker. The operational modes of an MS are described in details in [Fad10].

3.1 Work plan

The work on DTRAB is taking a systematic approach to detecting and tracing back attacks against encrypted protocols. We exploit learning from statistical data obtained during network operation to detect anomalous traffic, i.e., suspected attacks. A traceback function is used to stop malicious flow as early as possible. These three phases to realize this system and our approach to them are elaborated in this section.

As pointed out earlier, our work on DTRAB will focus on the *DTRAB Learning Phase*. To accurately identify anomalous behaviours, it is essential to study the protocol implementations and standard documents, such as RFCs, from which we can delineate the normal mode of the protocol operations. Additionally, the protocol behaviour in a network is subject to changes in different periods of a day. For instance, a corporate website may be accessed heavily during the day and less so at night. To reflect these factors, a statistical profile over time will be developed in the learning phase during the normal network conditions or near normal network conditions with low acceptable levels of suspicious activities. Each MS in the envisioned topology creates a database with features extracted from the non-encrypted headers of the monitored traffic over time. For identifying attacks, the MS monitors various protocol operations for anomalous modes of operation (e.g., number of failed sessions) and records them in such a manner that they serve as parameters to the non-parametric Cusum algorithm [ZRO$^+$07] used by the MSs in the detection phase. The format of a typical table from the database is shown in Figure 3. The two fields, S_n and T_n, which indicate the number of failed sessions and number of total sessions, respectively, are sampled over the profiling interval Δ_n. Using these parameters, the fraction of failed sessions, F_n, is then computed and stored in the database.

Moreover, our work covers the *DTRAB Detection Phase*. The detection approach adopted in DTRAB relies on detecting the point of change in the encrypted protocol behaviour as quickly as possible under an attack. For this purpose, we employ the non-parametric Cusum algorithm, which is a statistical tool. It is to be noted that the term "non-parametric" implies that the scheme may be adopted without having any knowledge of the traffic distribution beforehand (i.e., in a self learning fashion). We employ the non-parametric Cusum algorithm at the MSs to detect points of changes in the network behaviour at the advent of an anomaly. In [Fad10], there is an example of the proposed detection method analysing a random sequence of the number of failed SSH sessions, in a considered network flow, using a number of monitored profiles over equal profiling periods Δ_n.

The last step in our approach is the *DTRAB Alert/Trace Back Phases*. When the non-parametric Cusum algorithm detects an anomaly, the Cusum sequence begins to increase. Once it exceeds a determined threshold, the MS generates an alert to the server and the neighbouring MSs. The MS can also request the server to slow down the protocol response in an attempt to thwart intrusions such as remote timing attacks. Finally, the MS switches to the trace back mode to identify the attacker, which is described in details in [Fad10].

4 Self-Learning features in QoS^2

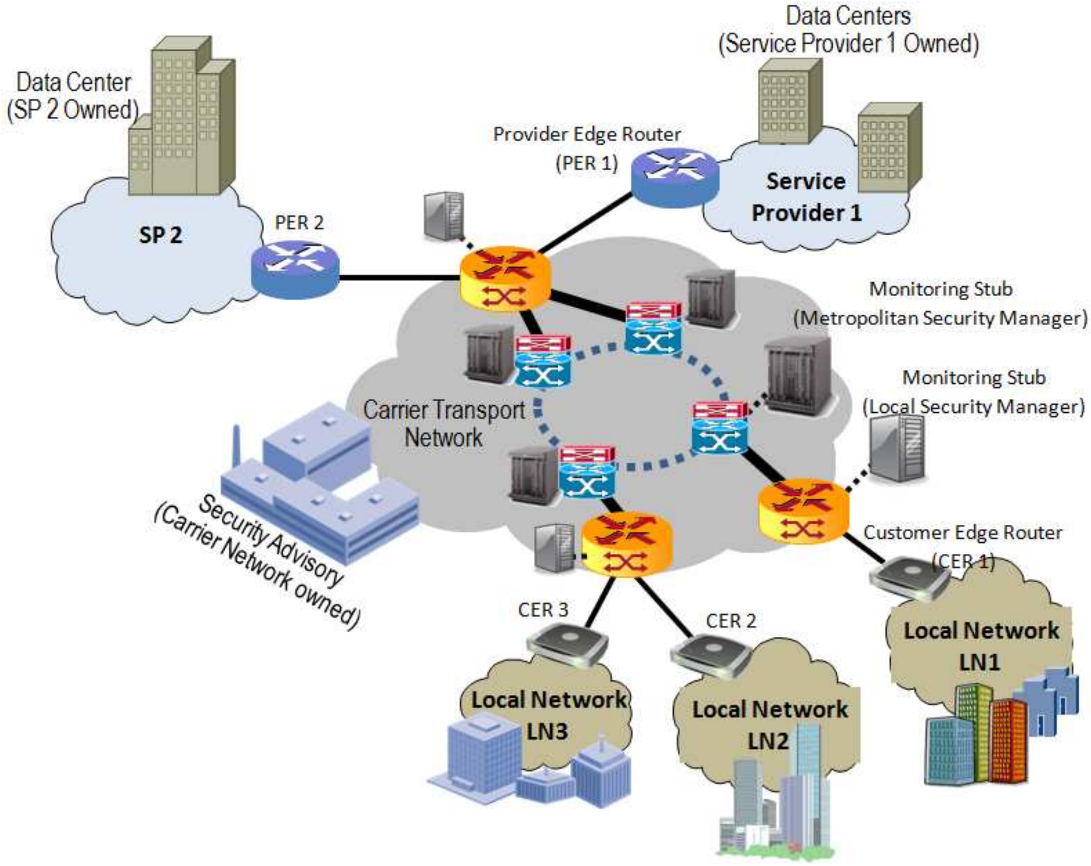


Figure 4: An example deployment scenario of QoS^2.

The QoS^2 framework – a framework to integrate Quality of Service with Quality of Security – has been introduced in Deliverable D2.1 [DGP$^+$10]. Figure 4 depicts an example deployment scenario of this QoS^2 framework [Tal10], portraying a carrier transport network administered by a particular Network Operator (NO) and connecting a number of content/service providers (i.e., clients of the network operator) to their customers, located in different local networks.

The overall network topology comprises a number of Monitoring Stubs (MSs) (i.e., similar in spirit to the DTRAB solution), which are intelligently deployed over the core network next to strategic routers. These MSs form a hierarchical threat detection system, consisting of two layers, namely Local Security Monitors (LSMs) and Metropolitan Security Monitors (MSM). Threat detection follows the following steps:

- The LSMs gather behavioural anomaly information of a particular local network and deliver them to the respective MSMs.

- Upon receiving information about a suspicious event from one of its LSMs, a MSM filters through its database of past attacks and evaluates if the threat is real.
- If the suspicious event is matched with one of the previous threats, the MSM notifies the security advisory.
- To avoid false alerts, the security advisory verifies if a similar threat originated from any other local networks administrated by another MSM.
- Once the security advisory judges the alert as true, it consults a built-in library of existing attacks and their counter-measures (to be developed based on learning, see the section below) to find the most appropriate mechanism to combat against the arising challenge.
- The security advisory defines then the threat level and relays the information pertaining to the threat detection and counter mechanism to all MSMs,.
- MSMs, in turn, forward the information to all collaborating LSMs.
- LSMs and MSMs finally communicate the security advisory instructions to the respective network elements that enforce them.

The security advisory also notifies the service managers of the different service providers of details on the on-going threat and triggers them to take adequate measures to adapt their QoS demands to the new network dynamics. Instructions on QoS adaptation/relaxation is communicated either to servers, to end users, or to both when required.

4.1 Work plan

Regarding the QoS^2 framework, there is still need to define a library that maps each specific scenario of a challenge type to a threat level and an appropriate defensive measure. Under a particular threat level, the same library characterizes a relevant defensive measure with a set of security related metrics, mapped to a corresponding set of QoS parameters that indicate the impact of the underlying defensive measure on the overall QoS. The next step in our QoS^2 work is to identify adequate learning tools for the said mappings. Indeed, two types of learning approaches are currently considered for evaluation.

In the first one, a straightforward one, the network keeps records of a challenge (see Section 2) that actually took place and maps it to the corresponding countermeasure (e.g., out of many launched) that exhibited the most optimal performance in terms of coping with the challenge while maintaining acceptable level of QoS (the *diagnosis* stage of the background control loop). Challenge details/symptoms and corresponding measures, along with insights on their overall performance, are then stored in the above mentioned library, for later use (*refinement* of the system). In the future, if the network encounters a similar challenge, the network refers to the constructed library to decide which measure with what granularity to deploy to cope with the challenge while maintaining the targeted level of QoS.

In the second learning approach, challenges are simulated and test runs are launched on a virtual sub-network of the network (e.g., with x% of resources out of the overall resources). The monitoring stubs (i.e., both local/metropolitan security monitors) quantify/record the impact of the challenge on the performance of the virtual sub-network (i.e., to define challenge symptoms) and the security advisory applies different measures with different granularities to assess how the virtual sub-network recovers from the challenge and how the QoS metrics are impacted (*diagnosis*). This assessment helps the security advisory to identify the optimal measure with the optimal granularity to tackle the simulated challenge while keeping a desired level of QoS (*refine*).

5 Learning to Improve Multi-path Forwarding Protection Schemes

In deliverables D2.3 [DSS+10], we introduced the wireless mesh backhaul context. In carrier-grade backhaul networks, especially in optical metro backhauls, traffic protection schemes are commonly used to prevent packet loss in case of link failures. We have already previously discussed various existing protection schemes such as path protection (PP), node protection (NP), link protection (LP) and segment protection (SP). Figure 5 gives an overview. We have also proposed a novel protection scheme called Rope-Ladder Protection (RLP) that has higher path diversity and path lifetime and can lead to smaller packet loss gaps than other schemes.

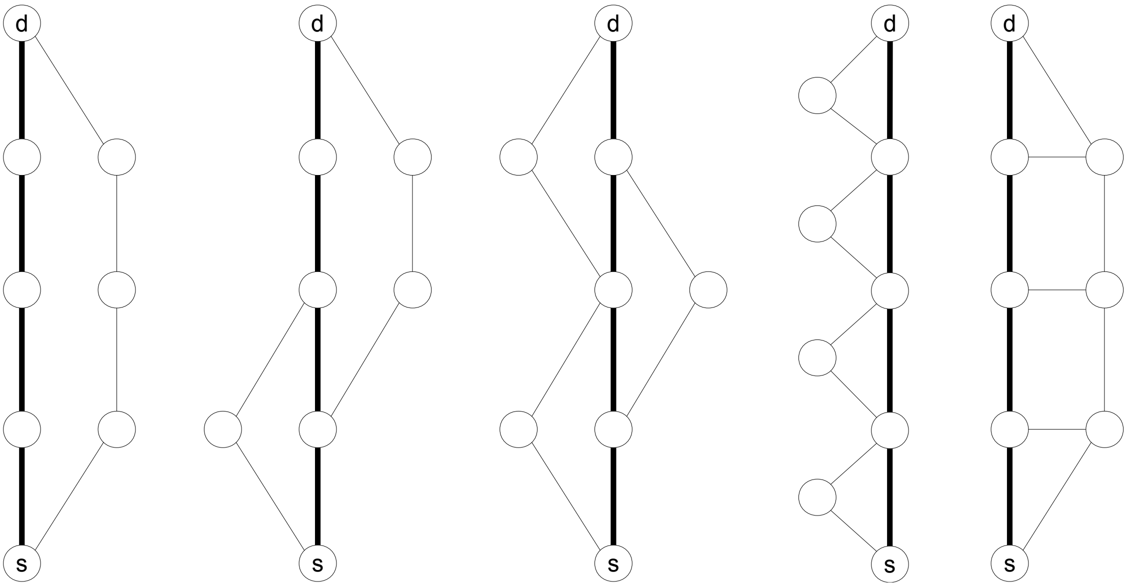


Figure 5: Different protection schemes. From left to right: PP, SP, NP, LP, RLP.

Generally, however, the selection of an appropriate protection scheme depends on several factors. There is no single best protection scheme for all possible scenarios. Among the decisive factors for scheme selection are the QoS requirements of the traffic to be protected, the length of the path between source and destination, the stability of the links, the amount of available resources, the mobility of the nodes, etc. While a first intuition of a good protection scheme for a given context could probably be derived based on a static model, it will be beneficial to introduce a self-optimization process for optimal scheme selection that is based on learning.

A second field where learning can be applied, is in the selection of an appropriate RLP *variant*. As was pointed out previously, it is not always possible to construct perfect rope-ladders (where every node on the primary path is connected via a rung to a node on the backup path, cf. Figure 6). This is the case, for example, in sparse networks which are not even bi-connected. Likewise, even if possible, it is not always necessary to build perfect rope-ladders. Perfect rope-ladders consume a lot of wireless resources (since even backup resources on rungs and backup paths must be reserved in carrier-grade backhaul networks) and this can actually be wasteful if the traffic is not overly QoS-sensitive or if link stability is

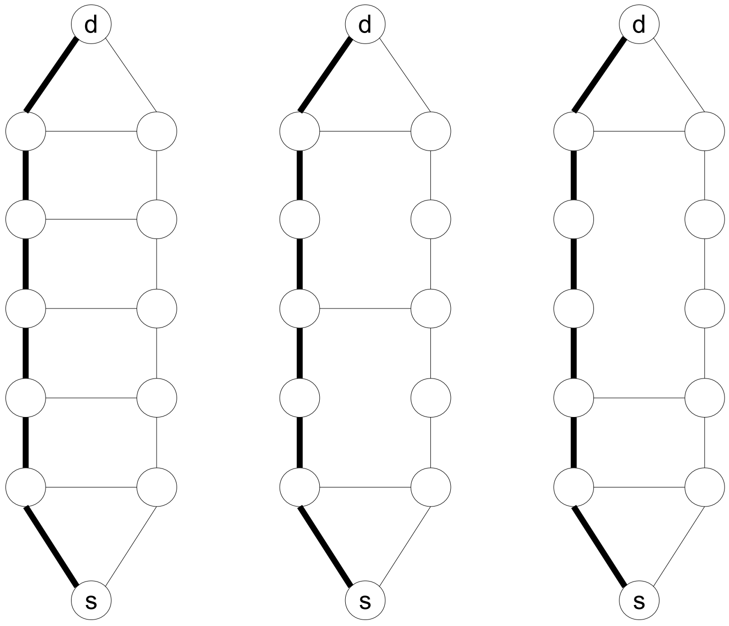


Figure 6: Different RLP variants. From left to right: perfect rope-ladder, regular rope-ladder, irregular rope-ladder.

very high. In such cases, backup resources would always be reserved, but almost never used. Consequently, RLP was defined in a way that allows to construct imperfect rope-ladders, i.e. ones where some of primary nodes are actually not connected via a rung to the backup path. The design of such imperfect rope-ladders can be so that only every second primary node has a rung connection, for example. We call the general case, where only every i-th primary node has rung connectivity, a regular rope-ladder. It could also be that the number of consecutive primary nodes which do not have rung connectivity is dynamically determined based on some metrics, such as the network connectivity in that network region. Such more dynamic variants are called irregular rope-ladders. Figure 6 depicts an example. Generally, learning can help optimize the decision for a rope-ladder variant that is optimally suited for a given specific context.

Depending on the network topology, links will generally not all be equally critical. Criticality can be understood in terms of link reliability (which can significantly differ in wireless scenarios), but also in terms of impact that the failure of a particular link has on the overall network robustness and load distribution. Given a set of link criticality values (determined, for example, by the Graph Explorer [DSS^{+}10]) for the primary links of a rope-ladder under consideration, it may actually not be necessary to protect the whole primary path with a backup path. Instead, depending on the criticality of these links, only certain subsections of the primary path may need protection. This can save backup resources in a way that hardly affects path reliability. It can also make it easier to construct RLP schemes at all in sparse networks with limited node degree. In this context, again, learning can be introduced to adjust the parameters in the Graph Explorer that steer the evaluation of link criticality values as well as the decision of whether or not to protect a given primary link.

One advantage of rope-ladder protection is that a rope-ladder can still provide a path from source to destination even after one or more links of primary and/or backup path have failed. Normally, the repair of a "damaged" rope-ladder would be started immediately after a failure of one of its links in order to avoid situations where further link failures lead to complete disconnections between source and destination. However, based on the criticality of the links in a certain rope-ladder subsection (as determined by the Graph Explorer), a better decision may be to defer repairing until a later point. This later point might be reached when either the criticality values have changed or additional link failures have led to a higher end-to-end risk for the rope-ladder as a whole. It is easy to see that in the latter case, for example, the amount of repair (signalling) overhead is reduced since one repair can be done to address multiple failures at once. However, the decision whether and until when to defer a repair is actually fairly tricky. We anticipate that learning can help here in order to optimize future deferral decisions based on previous experiences. Specifically, if deferrals have turned out to be too late for a certain context, parameters in the decision process will be adapted such that repairs are made earlier in the future.

Regardless of the application area for learning in the domain of traffic protection, it is important that adaptation of our protection scheme is always possible in two directions. For example, in the case where we want to adjust the amount of rung connectivity between primary and backup paths, it must be possible to increase as well as decrease the level of connectivity based on previous experiences. Probably, the more trivial part in this example is to increase the number of rungs once a problem with the rung density has been detected (e.g. too much packet loss or too large loss gaps). The more difficult part is how to decide whether to decrease the connectivity in case no problem with the current connectivity has occurred. One solution to this problem is related to strategies used for modulation rate adaptation in wireless radios. There, whenever the bit error rate gets too high to be further tolerated, the modulation rate is decreased. On the other hand, the rate adaptation algorithm constantly tries to increase the modulation rate as long as the bit error rate is sufficiently low. A similar strategy could be applied to rung connectivity adaptation.

5.1 Work plan

In the immediate future, we are planning on studying the introduction of learning into the protection scheme construction process. We will thereby focus on a subset of the above areas. The different areas of learning outlined above impose certain requirements on the data that needs to be collected. Important measures are packet loss rate and the size of loss gaps. The frequency with which these measurements are propagated to the centralized path computation element depends on the application. For online adaptation of existing protection schemes, such measurements need to be sent regularly, at least event-based, i.e. in case loss rates cross certain thresholds. The DISco API (see Section 2) is evaluated for its suitability of forwarding this information.

For updating the path computation function to adjust the construction parameters depending on its operational success or to recommend a different protection scheme for the next request (*refinement*), overall loss statistic at the end of the lifetime of the multi-path structure are required. Another item that must be kept track of is the involved links in a multi-path construct and link failure events with time stamps, to be able to obtain knowledge about the optimal timing for repair. Our plan is to study which additional data needs to be obtained for different of the above areas. Assessing the various information sources features the *diagnosis* stage of our strategy. In each case, that data will probably be aggregated in a centralized entity, since path computation in our carrier-grade wireless mesh network scenario is assumed to be centralized. This also nicely fits the fact that the Graph Explorer, which might assist the path computation process, is also a centralized entity.

6 Using machine learning for virtual service migration

Virtualization of services offers novel resilience mechanisms. By abstracting from the hardware, a virtual service can be migrated from one hardware platform to another. This enables a dynamic reaction to challenges like the destruction of hardware, an impaired communication environment, or unusual but legitimate requests for service. Depending on challenge detection and analysis methods, a resilience manager implementing virtual service migration can take challenge parameters and decide how to best react to a given challenge. As indicated in [FFAH10], virtual service migration is composed of a set of distinct actions. In order to maximise the resilience of a virtual service migration it is necessary to govern the migration, taking into account the effects different actions have on service resilience.

When using service migration as a resilience mechanism, one has to be careful to actually improve the situation, not make it worse. There are obvious parameters to consider when carrying out a migration. For example, the target platform should not be affected by the challenge to be countered. If the challenge is an upcoming power failure throughout a whole datacenter, it does not make sense to perform service migration within the datacenter. Moreover, the target platform should have the capabilities to support the migrated service. E.g., if a lot of services are migrated to the same target platform, QoS of each single service is likely to suffer significantly. Requirements like this can be checked before carrying through the actual migration. However, there may be side effects that are not so obvious with regard to their influence on service performance. For example, a service may actually have implicit assumptions about its location in the physical network topology. Moreover, services can be interdependent in a non-obvious way, possibly causing disruption when the physical location of one service changes. An example of this would be a web service that depends on a good connection to a database server in order to keep up with incoming requests. If the service is migrated due to a challenge, although the change of topology can be masked to some extent, the quality of the connection to the database server is likely to drop, reducing overall service quality.

These problems are hard to identify before service migration takes place. Moreover, implicit dependencies within the system may actually trigger an unexpected degradation during or after a service migration. Finally, some challenge parameters may differ from the analysis provided via the challenge detection mechanisms. For example, the lead time until a challenge affects the service in question might be shorter (or longer) than predicted. A learning framework would try to evaluate a service during and after a migration and identify problems that have been neglected when deciding to migrate. This information can then be used to better understand the implications of further service migrations and help to improve the migration strategy when facing the next challenge. By applying these concepts it becomes possible to gradually improve the quality and resilience of virtual service migration.

In order to better understand the impact of migration strategies, a viable first approach is to create and investigate an appropriate model of virtual service migration. This will allow the comparison of different migration strategies in a formal way, regarding their resilience. In addition, it is necessary to develop and apply metrics that allow to judge the resilience of a given situation. Information gathered in real situations can then be fed back into the model, further refining the results.

Attempting to achieve the goals mentioned above, we take the following points as objectives for our work in this task.

- Improving the resilience mechanism of virtual service migration requires to first understand how hidden parameters can influence service quality and performance. We want to create a formal model of virtual service migration that will allow us to identify those hidden parameters and to better understand the impact of different migration strategies.

- In order to inform a learning framework, it is necessary to find a universal way to judge service quality and performance during and after service migration. This requires elaborate service monitoring mechanisms. We therefore want to investigate interoperation between monitoring mechanisms and service virtualisation
- We want to develop appropriate service resilience metrics that allow us to calculate the resilience of a given system state. This will help us to understand how a service is going to behave in a certain challenge scenario.

Taking this together, the implemented resilience strategy can be evaluated and information on how to counter future challenges can be gathered.

6.1 Work plan

As a first step, it will be necessary to identify an interface via which a resilience manager for virtual service migration could be gradually and autonomically improved, leading to a *refinement* of the resilience strategy. This involves the integration of elaborate measurement and monitoring components within the controlled machines, as well as interfaces for information dissemination within the network. DISco (see Section 2) will be evaluated as a possible solution for dissemination of monitoring information.

We also plan to create a formal model of virtual machine migration in order to get quantitative information about the resilience of migration strategies. This will require finding an appropriate abstraction of virtual service migration and encoding it in a formal model. Moreover, a set of metrics relevant for virtual service migration has to be defined and applied within the model. This will be part of the *diagnosis* of our strategy. The information gathered with this model will then be compared and validated with our WP4 experiments in large-scale testbeds. To this end, we are currently working on an implementation of virtual service migration within the German G-Lab testbed facilities.

7 Conclusion

In this deliverable we have outlined the ResumeNet approach to design and implement the background control loop of our general strategy. This control loop consists of two stages: first, a *diagnosis* stage analysis the performance of past cycles of the realtime control loop. Second, the refinement stage evolves the system to a better operational state for future operation by autonomously modifying configurations or policies and by the human operator introducing new features and mechanisms to the system. In order to gain a better understanding of the information which has to be made available to the diagnosis stage, we investigate five use case scenarios from different domains encompassing defensive measures, challenge detection and system remediation and recovery mechanisms – mechanisms from all four stages of our realtime control loop.

The findings of these investigations will be reported in the updated version of this document (D2.4b) at the end of the ResumeNet project.

References

[DGP+10] C. Doerr, E. Gourdin, G.G. Popa, J. Omic, J.P.G. Sterbenz, J.P. Rohrer, T. Taleb, and P. Van Mieghem. D2.1b: Defensive measures for resilient networks. ResumeNet Deliverable, August 2010.

[DSS+10] C. Doerr, M. Schöller, P. Smith, N. Kheir, J. Lessmann, and C. Rohner. D2.3b: Remediation, recovery and measurement framework. ResumeNet Deliverable, August 2010.

[Fad10] Fadlullah Z., Taleb T., Vasilakos A., Guizani M., and Kato N. Dtrab: Combating against attacks on encrypted protocols through traffic-feature analysis. *IEEE/ACM Transactions on Networking*, 18(4):1234–1247, Aug. 2010.

[FFAH10] A. Fessi, A. Fischer, and Y. Al-Hazmi. D3.3: P2P Overlays and Virtualization for Service Resilience. ResumeNet Deliverable, August 2010.

[JEMP09] W. Jouini, D. Ernst, Ch. Moy, and J. Palicot. Multi-armed bandit based policies for cognitive radio's decision making issues. In *3rd International Conference on Signals, Circuits and Systems*, pages 1–6, Nov 2009.

[LSZB10] J. Lessmann, M. Schller, F. Zdarsky, and A. Banchs. Rope Ladder Routing: Position-Based Multipath Routing for Wireless Multi-Hop Networks. In *2nd IEEE WoWMoM Workshop on Hot Topics in Mesh Networking*, June 2010.

[MSF10] S. Martin, P. Smith, and M. Fry. D2.2b: New challenge detection approaches. ResumeNet Deliverable, August 2010.

[Nas09] Mohamed Nassar. *VoIP Networks Monitoring and Intrusion Detection*. PhD thesis, Henri Poincar'e University - Nancy 1, 2009.

[SS09] M. Schöller and P. Smith. D1.1: Understanding challenges and their impact on network resilience. ResumeNet Deliverable, August 2009.

[Tal10] Taleb T., Hadjadj-Aoul Y., Benslimane A. Integrating security with qos in next generation networks. In *IEEE Globecom*, Miami, USA, Dec. 2010. IEEE Computer Society.

[ZRO+07] Qi Zhang, Carlos Rendon, Veronica Montes De Oca, Prof. Daniel R. Jeske, and Dr. Mazda Marvasti. A nonparametric cusum algorithm for timeslot sequences with applications to network surveillance. *High-Assurance Systems Engineering, IEEE International Symposium on*, 0:435–436, 2007.