A Generic Model and Architecture for Automated Auditing

Hasan[1], Burkhard Stiller[1,2]

[1] Computer Engineering and Networks Laboratory TIK, ETH Zürich, Switzerland
{hasan, stiller}@tik.ee.ethz.ch
[2] Computer Science Department IFI, University of Zürich, Switzerland
stiller@ifi.unizh.ch

Abstract. Research has been performed in areas of auditing, a.o. security auditing, compliance auditing, financial auditing. In order to increase the efficiency of and to allow for continuous auditing, auditing tasks must be automated, which is only possible if audit data are available digitally and suitable algorithms exist. Different areas of auditing follow different objectives, thus require different detailed tasks to be performed, yet they share a common auditing model. This is based on the consideration that in general auditing deals with the evaluation or examination of facts against a set of compliance specifications. The objective of this paper is to develop a generic model and architecture for automated auditing, thus providing the basis for the development of auditing work for specific applications. To show its general applicability, the proposed model is applied to different areas including Service Level Agreement (SLA) compliance verification and Intrusion Detection Systems. A full-fledged example is discussed showing in detail how the generic architecture is applied to the SLA compliance verification.

1 Introduction

Auditing is a widely applied concept for investigating the adequacy of a system against a set of requirements. Traditional areas of auditing comprise amongst others financial audits, compliance audits with respect to laws and regulations, performance audits, and quality audits. The wide use of the Internet by research and government institutions, companies, and individuals, as well as the commercialization of Internet services have opened up a new and important area of auditing including information system security audits and Service Level Agreements (SLA) compliance audits.

The development and deployment of the Internet has bridged the path to the information era, where information becomes a vital resource. Usually information, in form of digital objects, is stored in a computer system, which is connected to the Internet. Since access to the Internet can be gained by anyone, the Internet is subject to attacks. To cope with attacks, various Intrusion Detection Systems (IDS) have been developed. An IDS is a security audit tool to reveal unauthorized access attempts.

Today, many business relationships between a service provider and a customer are formally defined in terms of SLAs, which specify contractual commitments of a provider on which services and with which measurable quality level the provider will fur-

nish [4]. The committed quality level of a service is specified in a set of Service Level Objectives (SLOs) in form of service metrics, threshold values, and tolerances [16]. Failure to perform contractual obligation is a contract violation. In order to detect this violation all related business transactions have to be accounted for and audited. Furthermore, for Internet services many communication protocols have been standardized and policy-based network management systems have been developed, where auditing can be useful to find non-compliances in protocol implementations and policy decisions.

Traditionally, auditing is accomplished by human auditors through a manual audit of paper documentation, which is very time-consuming. This leads to the question whether, at least, parts of the auditing process can be automated. Automation is easier to achieve in those areas of auditing, where materials are available in a structured digital format. Auditing deals with varieties of information, which often have a complex interrelation. In many cases, auditing requires a strong analytical skill of the auditor, and the transfer of this skill to an automated auditor poses a challenge. The two main reasons for automated auditing are a high degree of efficiency in processing a large number of audit data and earlier detection of non-compliances through continuous auditing.

Thus, the key goal of this paper is to develop the common denominator of auditing tasks in different areas and to design a generic model and architecture for automated auditing. Based on this model and architecture a framework for various auditing purposes is derived to minimize further efforts in implementing specific auditing applications.

The remainder of the paper is organized as follows. Section 2 discusses related work from different auditing areas. While Section 3 presents a generic auditing model, Section 4 develops the generic architecture. The application of the model and architecture is presented in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Most of existing approaches in auditing are dedicated to specific objectives. Although some proposals show to a certain extent a general approach, a generic auditing model and architecture—to the best knowledge of the authors at the moment of writing—is not yet available. Hence, this section describes major related work in specific areas of auditing, including security auditing and SLA compliance verification.

IDSs have been a research area of security auditing since the beginning of the 1980s. In 1987 [5] presented a model for a real-time intrusion detection expert system, which provides the basis for many IDSs. The model can be regarded as a rule-based pattern matching system and contains the following six main components: subjects, objects, audit records, profiles, anomaly records, and activity rules. In this model actions performed by subjects on objects are essential, which is valid for an IDS, but not in all areas of auditing. For example, in the case of SLA compliance verifications services delivered by objects to subjects are relevant. Therefore, the relation between subjects and objects is not described in a generic auditing model. Furthermore, profiles in this model, which describe a normal behavior of subjects on objects, are updated based on audit records. Here, profiles define the specifications to be met. However, general auditing does not modify compliance specifications so that they are met by the normal behavior.

The architecture of a general IDS comprises of event generators within a target system, analysis engines, and a response unit [11]. Components can be distributed; analysis engines can form a hierarchy. Recent architectures use autonomous and mobile agents to perform the task of event generators and analysis engines. Distribution of IDS components happens within a single administrative domain, whereas general auditing can stretch across multiple ones.

Academia and industries have performed research and developed prototypes or even products for SLA management and monitoring of SLA compliances. However, most of these approaches are dedicated to specific services, e.g., web services, or a certain set of SLA parameters, e.g., availability, round-trip time, and response time [3], [6], [7], [9], [10], [15]. The IBM's Web Service Level Agreement (WSLA) Framework shows a general concept for SLA management and defines a language to specify SLAs, focusing on web services [10]. It defines five stages of the SLA management lifecycle: negotiation and establishment, deployment, measurement and monitoring, corrective management action, and termination. The functionality needed for these various stages is implemented as WSLA services, which are intended to interact across domains. The SLA Compliance Monitor comprises of three WSLA services: SLA Deployment, Measurement, and Condition Evaluation. However, in order to be a generic auditing framework SLA specific elements in the WSLA Framework need to be generalized.

In other more traditional areas of auditing, continuous efforts are done in providing software tools and expert systems to aid human auditors in carrying out the audit. There are approaches to automate certain tasks of financial audit, e.g., through the use of Artificial Intelligence [18], electronic audit data warehouses, and data marts [12]. In this area, audit software tools are utilized to help a human auditor in the analysis of transactional data [1], [2]. However, human interventions are still needed.

3 A Generic Model for Automated Auditing

The different areas of auditing follow different auditing objectives and have different tasks and characteristics, but they share a common model. A general model for automated auditing requires a general definition of auditing. The definitions given by the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and the Internet Engineering Task Force (IETF) focus on auditing in the area of security [13], [14]. Both define the term Security Audit in a similar way. The U.S. Committee on National Security Systems defines the term Audit instead of Security Audit [17]. Although this definition does not emphasize security, several other auditing areas are not covered. Therefore, a concise, general definition of Audit is given: "An Audit is a systematic and independent examination of facts on system activities to determine the degree of compliance with a pre-defined set of specifications." Auditing is the process of conducting an Audit, and an Auditor is an entity that carries out the Audit.

Fig. 1 depicts a class diagram of the generic auditing model in the UML (Unified Modeling Language) notation. The model consists of 8 classes divided into 3 roles, i.e., Auditor, Auditee, and Accountant, and 5 data, i.e., Audit References, Compliance Specifications, Activities, Facts, and Audit Reports. Compliance Specifications are a set of

specifications derived from laws or regulations, contracts or agreements, pre-established policies, and procedures, which the particular system under audit, i.e., the Auditee, has to follow. Note that "follow" means either to meet a specification to achieve an expected state or to avoid meeting a specification to not reach an unexpected state.

An Auditee carries out Activities to achieve a certain goal. The goal itself is irrelevant, however, it is expected that in performing those Activities the Auditee follows the Compliance Specifications. Therefore, those Activities are observed by an Accountant, who records and stores them as Facts. Facts about such Activities reveal whether the targeted Compliance Specifications hold. A Fact is, therefore, a piece of information presented as having an objective reality. A Fact can be accompanied by an Evidence which furnishes a proof, i.e., that ascertains the truth of a matter, thus, increases the informational reliability of a Fact. Evidences are obtained technically through non-repudiation mechanisms, which can be used, e.g., to prove service consumptions [8].

An Auditor conducts an Audit by evaluating Facts based on related Compliance Specifications to detect violations. An Audit has to be conducted according to valid procedures, standards, laws, or regulations, being termed Audit References. Hence, Audit References define the legal or generally accepted way to conduct an Audit in a particular area of auditing. While an Auditee has to follow Compliance Specifications, an Auditor has to follow Audit References. Thus, activities of an Auditor can be subject to the Audit by another Auditor. The Auditor generates an Audit Report based on the result of the Audit. This report can be consulted by the Auditee to avoid further violations.

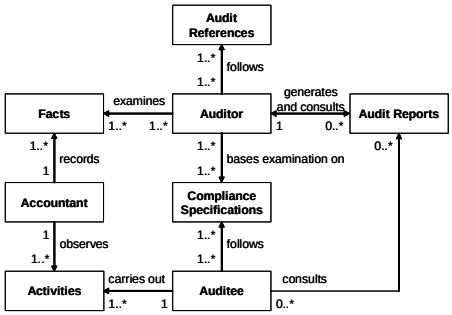

Fig. 1. Generic auditing model

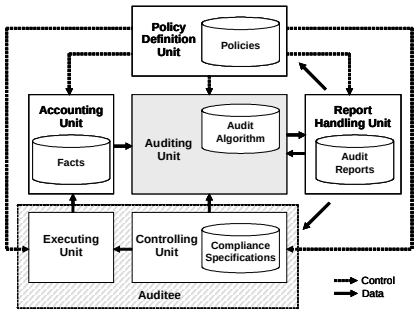

Fig. 2. Generic auditing architecture

4 Architecture

Driven by the general model, the generic auditing architecture is designed, which is applicable to different auditing areas. Architectural components can be distributed across domains, and a suitable approach for an efficient Automated Auditor is included.

4.1 The Design of the Generic Auditing Architecture

Fig. 2 depicts the generic auditing architecture having the Auditing Unit as a major component, which contains a set of Automated Auditors. Automated Auditors may in-

teract with each other, if required, in conducting a particular Audit. The Auditing Unit implements the Audit Algorithm of a particular auditing application. An Audit Algorithm is a technical description of Audit References.

An auditing process requires at least two types of inputs: Compliance Specifications and Facts. Facts describe what actually happens, whereas Compliance Specifications describe expected situations. An Auditee should follow Compliance Specifications in carrying out Activities, hence, an Auditee is divided into a Controlling Unit and an Executing Unit. The Controlling Unit defines Activities to be carried out by the Executing Unit and makes sure that the Compliance Specifications are held. The Controlling Unit provides for Compliance Specifications, whereas the Accounting Unit delivers Facts about Activities performed by the Executing Unit to the Auditing Unit.

The result of an Audit is a report with statements on the degree of compliance of the Auditee's Activities with respect to pre-defined Compliance Specifications. In certain cases, results of previous Audits serve as an input to the current Audit, therefore, the flow of Audit Reports between Auditing Unit and Report Handling Unit is bi-directional. The Report Handling Unit is responsible for maintaining Audit Reports.

The generic auditing architecture uses a policy-based approach to configure and to control the behavior of different units. The policy-based approach offers the advantage of being able to separate decision taking instances from executing instances, hence, it allows for a modular structure. Of course, different modular decision systems may be applied as well. Policies are defined and managed by a Policy Definition Unit. As an example, Audit Policies can be defined to influence the behavior of an Auditing Unit in conducting an Audit without violating Audit References.

In many cases, if the result of an Audit reveals violations to a particular Compliance Specification, a pre-defined corrective action needs to be taken. The action is carried out either by the Report Handling Unit, Policy Definition Unit, or Controlling Unit. Therefore, Audit Reports are also accessible by the Policy Definition Unit and Controlling Unit (cf. Fig. 2). There are three possible causes of a violation: (1) Inappropriately set up Audit Policies, (2) Imprecise Compliance Specifications, and (3) Executing Unit did not perform as expected. In general, cause (1) and (2) should not happen in normal operation, but can happen during learning or experimenting phase. Hence, the generic architecture foresees 3 different feedback control mechanisms to cope with violations:

1. Feedback to the Auditor through changing Audit Policies. This is useful, when violations have occurred unexpectedly due to Audit Policies being inappropriately set up. The Policy Definition Unit must be able to decide whether an Audit Policy is appropriately set up or not, which is a difficult task without human intervention.
2. Feedback to the Auditor through modifications of Compliance Specifications. This is useful, when violations have occurred unexpectedly due to imprecise Compliance Specifications. Here, the Controlling Unit must be able to decide, whether a Compliance Specification is precisely specified or not, which is also a difficult task without human intervention.
3. Feedback to the Executing Unit by the Controlling Unit through reconfiguration. This should be the case if the Executing Unit did not perform as expected.

Obviously, the use of one mechanism does not preclude the use of other mechanisms at the same time, because each mechanism serves a different purpose.

4.2 Distributed Architecture across Administrative Domains

In some areas, e.g., SLA compliance verification, auditing is applied in a multi-administrative domain (AD) environment. The application of the generic auditing architecture is not restricted to a single AD, instead, architectural components can be distributed across several ADs. However, such a distributed architecture requires trust among ADs.

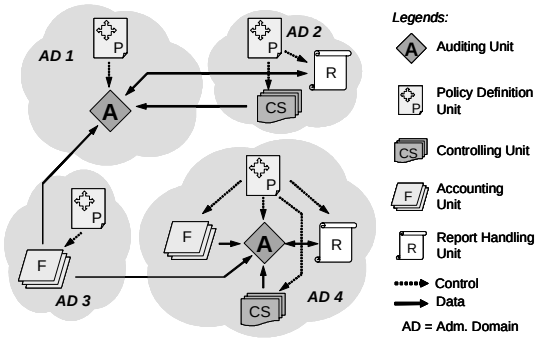

Fig. 3. Multi-administrative domain architecture

An AD can offer an accounting or auditing function as a service to other ADs. In Fig. 3 the AD 1 provides an auditing service, whereas the AD 3 offers an accounting service. The AD 2 uses the auditing service of AD 1, who further makes use of the accounting service of the AD 3. Here, AD 4 implements all the functions, but still needs additional accounting information from the AD 3 to conduct the Audit. Note that the Executing Unit and the feedback arrows are not shown to avoid overloading.

4.3 Automated Auditor's Internal Architecture

As described, an Auditing Unit contains a set of Automated Auditors and it implements the Audit Algorithm of a particular auditing application. This means that Automated Auditors have the task to execute the Audit Algorithm. In order to reduce implementation complexity and to achieve modularity, the following assumptions are made in designing the architecture of an Automated Auditor:

- An Auditing Unit deals with a set of Compliance Specifications. Without loss of generality each Automated Auditor is assumed to be responsible for a particular Compliance Specification.
- Each Compliance Specification contains a set of Compliance Conditions linked by a logical expression. Hence, the result of the evaluation of each condition as well as the evaluation of the logical expression linking all the conditions determine the compliance of relevant Facts with a Compliance Specification.
- A common Audit Algorithm exists which is valid for most auditing applications.

The approach developed here proposes the following common Audit Algorithm:

1. Interpret and apply valid Audit Policies during the Audit.
2. Interpret the assigned Compliance Specification, for which the Automated Auditor is responsible.

3. Retrieve relevant Facts and Audit Reports based on the Compliance Conditions.
4. Evaluate Facts and Audit Reports whether they meet Compliance Conditions. Evaluate the logical expression linking all Compliance Conditions.
5. Generate a report as a result of the evaluation.

Fig. 4 shows the architecture of the Automated Auditor to execute the proposed Audit Algorithm. The Policies Interpreter (PI) takes policy decisions and configures other components based on Audit Policies. The Compliance Specification Interpreter (CSI) retrieves the Compliance Specification assigned to the Auditor based on the configuration information from PI. It instantiates Compliance Condition Evaluators (CCE) and gives each CCE a Compliance Condition to evaluate. Each CCE subscribes relevant types of Facts and Audit Reports from the Facts Dispatcher (FD) and the Report Dispatcher (RD), respectively.

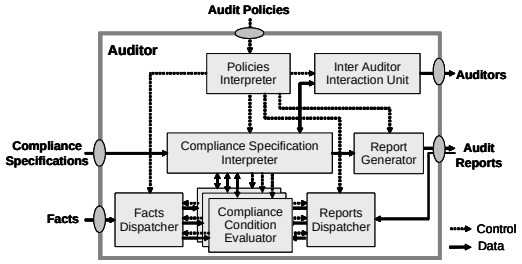

Fig. 4. Automated Auditor's internal architecture

FD is responsible for retrieving, filtering, and distributing Facts to the respective CCE. A similar function is assigned to RD. Each CCE examines received Facts and relevant Audit Reports in evaluating the condition given by the CSI and sends the result of this evaluation to the CSI. The CSI determines whether there is a violation of the Compliance Specification based on the result of each CCE. The decision of the CSI is sent to the Report Generator which is responsible for composing the Audit Report in a pre-defined format and storing it into a pre-configured location. If interactions among Automated Auditors are required to conduct a particular Audit, interactions are handled by the Inter Auditor Interaction Unit (IAIU). This requirement is stated either in Audit Policies or Compliance Specifications. In both cases the CSI is generally involved.

5 Application of Auditing Model and Architecture

The model developed and the generic architecture designed are applicable to various auditing areas, especially, SLA compliance verification.

5.1 Application of the Model

To start with the verification, Table 1 maps each generic class of the diagram in Fig. to different entities of three different auditing areas: SLA compliance verification, intrusion detection, and financial auditing.

Table 1. Application of the generic auditing model to different auditing areas

Model	SLA Compliance Verification	Intrusion Detection	Financial Auditing
Auditor	SLA Compliance Auditor	Intrusion Detection Engine	Auditor
Auditee	Service provisioning entities	Users of network infrastructures and services	Company, including its Accountants
Accountant	Meter and accounting entities	Sensors (usage monitoring and logging entities)	Company's Accountants
Audit References	Agreed procedures to conduct an Audit	Intrusion detection methods	Auditing standards, *e.g.*, Generally Accepted Auditing Standards (GAAS)
Compliance Specifications	Service Level Agreements	Signature-based: intrusion rules, patterns (signatures) vs. anomaly-based: heuristic rules, normal states or behavior	Accounting standards, *e.g.*, Generally Accepted Accounting Principles (GAAP)
Activities	Service deliveries	Usage of network infrastructures and services	Accounting of business transactions, in particular generation of financial statements
Facts	Meter data or accounting data, and event logs	Network traffic data and event logs	Financial statements
Audit Reports	SLA violation reports	Intrusion reports and alarms	Audit reports

5.2 Application of the Architecture: SLA Compliance Verification

To outline the application of the generic architecture to the Internet, the SLA compliance verification is taken. An SLA contains amongst others: customer ID, subscription start date, subscription end date, subscribed services and service classes, prices, SLOs, remedies in case of SLA violations, and limiting conditions. Information in an SLA is used for access control, meter configuration, and SLA compliance verification.

The following example in Fig. 5 provides a basis for the detailed description on the application of the generic architecture to SLA compliance verification. Provider P is a network operator providing QoS-enabled network services to her customers. The provider's network is connected to the Internet via two Border Routers (BRs), and customers can access the Internet via one of the three Access Routers (ARs). Provider P offers different network service classes to meet different customer and application needs.

The throughput parameter is part of SLAs between Provider P and her customers, and this parameter shall be guaranteed. In order to be fully precise in the definition of this guarantee, downlink and uplink traffic as well as incoming and outgoing traffic need to be distinguished (Fig. 5). The traffic coming from a terminal is the uplink traffic of this terminal and the traffic going to a terminal is the downlink traffic. Traffic entering the network is the incoming traffic, whereas traffic leaving the network is the out-

going traffic. Based on the above description two kinds of throughput guarantee can be defined: downlink and uplink throughput guarantee. Each guarantee is held in an SLO.

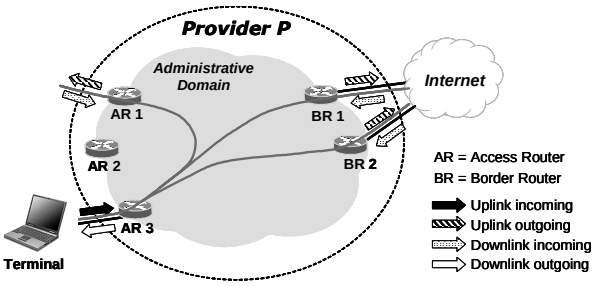

Fig. 5. Different traffic for throughput definition

Downlink throughput guarantee is defined as follows: "The percentage rate reduction in downlink traffic of a network Service Class between the incoming and outgoing traffic will be within dR_d (downlink rate reduction tolerance) in every pre-defined Metering Interval (MI) during the whole session. The rate of the downlink incoming traffic used in the calculation is at most equal to the downlink committed rate Rc_d." This guarantee defines a condition which can be expressed mathematically using the formula:

$$1 - \frac{Ro_d(t)}{Min\,(Rc_d, Ri_d(t))} < dR_d \quad . \tag{1}$$

In (1) $Ro_d(t)$ is the downlink outgoing rate, whereas $Ri_d(t)$ is the downlink incoming rate. Values of Rc_d, dR_d, and *MI* are determined by the service class. Uplink throughput guarantee is defined in a similar way.

Executing Unit. Service provisioning entities are the provider's network infrastructure, in particular QoS-enabled routers and switches. They must deliver services according to SLAs, hence, they determine the Executing Unit.

Controlling Unit and Compliance Specifications. Provider P defines and manages SLAs. She configures and operates service provisioning entities. Therefore, Provider P represents the Controlling Unit. In the area of SLA compliance verification, Compliance Specifications (CS) are derived from concluded SLAs and Compliance Conditions (CC) within a CS are determined by the related SLO. Obviously, an SLA contains more information than needed for defining CSs. Here, it is sufficient if a CS comprises of SLO metrics, service class, and condition expression. As Provider P only provides a single service, namely a network service, it is not required to have the name "network service" as subscribed services in the CS. An example of a CS reads as follows:

Specification Number: 001,
Metrics: Downlink Throughput,
Service Class: A (Rc_d = 1 Mbps, dR_d = 0.05),
Conditions: 1 - $Ro_d(t)$ / Min(Rc_d, $Ri_d(t)$) < dR_d,

CS 001 is valid only for customers of service class A, but an explicit list of customer IDs in CS 001 is not needed due to the fact that this case does not deal with auditing of access control or intrusion detection, but with auditing of SLA compliance. Access control entities are responsible to ensure that only authorized customers can access services. Hence, recorded Facts are assumed to contain correct/authorized customer IDs.

CS 001 is a simple CS, where parameter Conditions comprise of a single relational expression. Other CSs can be very complex, in which parameter Conditions contain several relational expressions linked by logical operators, and the evaluation is triggered by the occurance of some other Facts.

Accounting Unit and Facts. A meter is deployed in each AR and BR to capture the data rate of customers' traffic. To allow for rate measurements of different service classes, the traffic of each service class must be marked accordingly, e.g., by using Differentiated Service Code Points (DSCP). For the examination of downlink throughput guarantees the following information must be collected by meters: router interface, DSCP, destination IP address, meter interval start time, meter interval end time, and volume. The router interface determines whether the traffic being metered is an outgoing or incoming traffic. The DSCP determines the service class, i.e., the agreed Rc_d and dR_d. The destination IP address represents the customer of the downlink traffic. Time interval and volume are used to calculate the traffic rate.

To ease auditing, related meter data need to be pre-processed by accounting entities. This pre-processing includes data aggregation from different meters and information mapping. In case of downlink throughput guarantee, the resulting accounting record contains the following information: customer ID, service class, meter interval end time, downlink incoming rate, and downlink outgoing rate. Obviously, meter data and accounting data represent Facts about service deliveries and their quality level. Hence, meters and accounting entities build up the Accounting Unit of the generic architecture. To avoid manipulation of meter data or accounting data by provider (or customer) the measurement and accounting task can be carried out by a third party.

Auditing Unit: Automated Auditors. Automated Auditors have the task to evaluate whether the accounting data meet the SLOs specified in SLAs. Auditors deal with a set of CSs and a large number of accounting data recorded during service usage by a large number of customers. Therefore, it is reasonable to run a set of Automated Auditors, where each is responsible for a single CS and a fraction of the accounting data. In the example with Provider P, one way to divide the accounting data is by grouping them based on any combination of SLO metrics, service class, and customer ID. For example, an Automated Auditor AA1 is assigned to evaluate downlink throughput of traffic of service class A addressed to customer with ID ranging from 101 to 200. This means, an Accounting Unit must be able to selectively distribute accounting data to each Auditor.

As already described, CSI determines the core component of an Automated Auditor. The CSI of AA1 interpretes CS 001 and instantiates a single CCE, since parameter Conditions in CS 001 consist only of one relational expression. The CCE configures the FD to retrieve customer ID, downlink incoming rate $Ri_d(t)$, and downlink outgoing rate $Ro_d(t)$ from accounting data with parameters: Metrics = Downlink Throughput, Service Class = A, and Customer ID between 101 and 200. If there is a Fact with $Ri_d(t)$ and $Ro_d(t)$ values which do not satisfy the specified condition, a violation report is created.

Report Handling Unit and Audit Reports. Based on the evaluation result of the Auditing Unit, the Report Handling Unit determines, which SLA is violated and what actions, if any, need to be performed. It also keeps Audit Reports in a pre-defined format for later use. In the example the violated SLA is found by matching the customer ID in the SLA with the customer ID of the violating accounting record.

Policy Definition Unit and Policies. Provider P configures various units and defines policies to be consulted by those units, hence, Provider P represents the Policy Definition Unit. An example for a configuration item is the communication interface of an Automated Auditor. An Automated Auditor communicates with a Controlling Unit, a set of Accounting Units, a set of Report Handling Units, and if required, other Automated Auditors. Which Accounting Units and Report Handling Units have to be contacted depends on which Facts and Reports are to be retrieved and where to store all resulting Reports. Configuration parameters include a URL, i.e., IP address or hostname, port number, and protocol. The following configuration example states that in order to obtain data on downlink throughput of class A traffic for all customers, the Accounting Unit running at host testbed_db.ethz.ch should be contacted via port 13000 using the Diameter protocol.

Metrics: Downlink Throughput
Service Class: A
Customer IDs: ALL
Accounting Unit: testbed_db.ethz.ch:13000:diameter

Audit Policies are useful to influence the behavior of an Automated Auditor. For example, a policy can be defined for the Automated Auditor AA1 to treat specific accounting records differently. Assume that Customer ID 113 is allowed to send traffic with a downlink committed rate 10% above the rate for service class A during the month April 2005. In this regard an Audit Policy for AA1 reads as follows:

```
if (Customer ID = 113 and Meter Timestamp within April 2005)
  then use Rc_d = 1.1 Mbps.
```

Policies for other units may also be defined, if needed. For example, a Meter Policy for the Accounting Unit can be specified to adapt metering intervals to network load, because the smaller the interval, the more meter data need to be transfered. However, the chosen interval may not lie outside of the range defined in the SLA.

6 Summary, Conclusions and Outlook

Although different auditing areas require different audit tasks, they share a common model, which is shown by developing a generic auditing model and mapping elements of this model to entities in different auditing areas. A generic auditing architecture has been designed based on this model and applied to SLA compliance verification.

The implementation of an auditing framework has been planned and a first effort to develop a Compliance Specification Interpreter is being made. This requires the definition of a general compliance specification language, which is currently being studied. The ultimate goal is to show that it is feasible to provide a generic framework for most areas of automated auditing. In this regard, the model and architecture must be supported with formalizations of various aspects of auditing, including algorithm, policy definition, compliance specification, fact representation, and interface definition between those architectural components. Additionally, the privacy concern in inter-domain applications needs to be solved, and a set of use cases needs to be analysed. These form the main part of further research work.

Concluding, the generic model and architecture provide a common and flexible basis for further development in various auditing areas, in particular security auditing, SLA compliance verification, and business auditing. The availability of an auditing framework based on this generic model and architecture is very important, as automated auditing becomes crucial in many business and security processes, and a fast as well as efficient automated auditing is essential. In turn, future efforts to implement the auditing for specific applications are reduced heavily with the help of this framework.

Acknowledgments

The work has been performed partially in the framework of the EU IST project Daidalos (FP6-2002-IST Contract No. 506997), where ETH Zürich has been funded by the Swiss Bundesministerium für Bildung und Wissenschaft (grant No. 03.0141). The authors would like to thank their project partners, especially Portugal Telecom Inovação and Fraunhofer-Gesellschaft FOKUS.

References

1. ACL Services Ltd.: ACL Tops 2004 Internal Auditor Software Survey. (2004)
2. CaseWare IDEA Inc.: IDEA: Product Profile. (2004)
3. Daidalos: A4C Framework Design Specification. Deliverable D341 (2004)
4. D'Antonio, S., Esposito, M., Gargiulo, M., Romano, S.P., Ventre, G.: A Component-based Approach to SLA Monitoring in Premium IP Networks. First Intl. Workshop on Inter-Domain Performance and Simulation, Salzburg (2003)
5. Denning, D. E.: An Intrusion-Detection Model. IEEE Transactions on Software Engineering, Vol. SE-13, No.2 (1987) 222-232
6. G-NE GmbH: Konzeptionsansatz: Qualitätssicherung in IT-Outsourcing-Projekten mittels einer unabhängigen Prüfinstanz. Confidential Document (2002)
7. Hasan, Stiller, B.: Auditing Architecture for SLA Violation Detection in QoS-Supporting Mobile Internet. IST Mobile and Wireless Comm. Summit, Vol. 1. Aveiro, Portugal (2003)
8. Hasan, Stiller, B.: Non-repudiation of Consumption of Mobile Internet Services with Privacy Support. IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (to be published), Montreal, Canada (2005)
9. Itellix Software: Wisiba: Datasheet. (2003)
10. Keller, A., Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management, Vol. 11, Issue 1 (2003) 57 - 81
11. Lundin, E., Jonsson, E.: Survey of Intrusion Detection Research. Technical Report 02-04, Department of Computer Engineering, Chalmers Univ. of Technology, Göteborg (2002)
12. Rezaee, Z., et. al.: Continuous Auditing: Building Automated Auditing Capability. Auditing: A Journal of Practice & Theory, Vol. 21 Issue 1 (2002) 147-163
13. Shirey, R.: Internet Security Glossary. IETF, RFC 2828 (2000)
14. Study Group on Communication Systems Security: Compendium of approved ITU-T Security Definitions. (2003)
15. Softek Storage Solutions Corporation: SOFTEK EnView: Datasheet. (2004)
16. Telemanagement Forum: SLA Management Handbook, V1.5. GB917 (2001)
17. U.S. Committee on National Security Systems: National Information Assurance Glossary. (2003)
18. Vasarhelyi, M.A.: Artificial Intelligence in Accounting and Auditing, Volume IV: Towards New Paradigms. (1997)