Automated compliance checking of z/OS against GSD 331 and DISA STIG

Bharathi Venkataramanapa, G.

Award date:
2012

Automated Compliance Checking of z/OS against GSD 331 and DISA STIG

Geetha Bharathi Venkataramanapa
(0785813)

August 19, 2012

Master Thesis

Eindhoven University of Technology
Embedded Systems
Security Group

In cooperation with:
IBM Netherlands

Supervisor:
Name: Prof. dr. Jerry den Hartog
Position: Assistant Professor
E-mail: j.d.hartog@tue.nl

Tutor:
Name: Dr. Nicola Zanonne
Position: Assistant Professor
E-mail: n.zannone@tue.nl

Supervisor at IBM:
Hans Schoone
E-mail: hans.schoone@nl.ibm.com

Supervisor at Manipal:
Name: Prof. dr. Manohara Pai M.M
Position: Associate Director (Innovation Centre)
E-mail: mmm.pai@manipal.edu

Contents

List of Figures

List of Tables

Abstract

For corporations, one of the most valuable resources is its data. This critical data should be securely managed and controlled, and simultaneously made available to other users who are authorized to use it. In an IT environment, data security is defined as protection against unauthorized access, transfer, modification, or destruction. Since mainframe handles over 70% of business critical, transactional, and customer data being stored in the mainframe, security within the enterprise becomes a matter of high concern [36]. Several Governmental and Non-Governmental regulations exist such as DISA STIG, HIPAA, PCI-DSS, ISO-27002/27001, and GSD 331 in order to ensure data security in organizations. Companies need to comply with several regulations to ensure data security and also to avoid consequences of non compliance. Compliance management is a process in which organizations operate in accordance with some regulations. The process of verifying whether or not a given system fulfils the compliance requirements of regulations is compliance checking.

Compliance checking can be done either manually or in an automated way. Organizations face significant challenges in compliance checking due to increase in number of regulations across various domains. The major challenges involved in compliance checking are:

- Increasing costs
- Multiple Regulations every year
- Lack of industry guidance
- Multiple domains

In general, manually checking compliance status of a system requires huge labour, time and is often error prone. As a result, companies continually strive to come up with various automated compliance checking approaches to assess compliance status of a system. Automating the compliance checking process helps to simplify the assessment process, reduce costs and potentially increase the confidence of assessments. An automated compliance checking process can also help in addressing the problem of assessing the compliance to multiple regulations. This thesis aims to propose a methodology for implementing an automated compliance checking process. The proposed methodology is used to implement an automated compliance checking process for z/OS, a mainframe operating system. Prior to implementation, the regulations are analysed to determine which rules can be automated for the underlying system. The regulations considered in this thesis are DISA STIG and GSD 331.

Acknowledgements

This thesis report is a result of the master project that I have done at IBM Netherlands. Completing this Master project has been a wonderful and often overwhelming experience. I have been very privileged to have undoubtedly the most intuitive, smart and supportive advisers anyone could ask for, namely Prof. dr. Nicola Zannone, Prof. dr. Jerry den Hartog and Hans Schoone. Upon completion of my master thesis, I would like extend my deepest gratitude to my family for their undying support, love, affection and guidance throughout my life. Also, I would like to thank everyone who has contributed to the successful completion of this assignment.

First and foremost, I would like to thank my supervisors, Prof. dr. Jerry den Hartog, Prof. dr. Nicola Zannone, and Hans Schoone without whose support and guidance, this work would not have been possible. I would like to extend my special thanks to Prof. dr. Jerry den Hartog and Prof. dr. Nicola Zannone for patiently reviewing my thesis several times and providing me with their valuable comments. I would like to thank my supervisor at IBM Netherlands, Hans Schoone for giving me an opportunity to work on such an interesting assignment and giving me the freedom to apply my ideas in the project. Working under him was not only challenging but also a nice experience altogether.

I extend my deepest gratitude to Prof. dr. Mark van den Brand, the program director for the Manipal-TU/e dual degree program, Prof. dr. Manohara M. Pai, Prof. dr. Jos Baeten, and Prof. dr. Radhika M Pai for giving me an opportunity to do my masters in this prestigious institute. I would like to thank Prof.dr. Manohara Pai for being my project supervisor from Manipal University and for his valuable evaluation. I would like to thank Prof. dr. Mark van de Brand for being a part of my assessment committee. I am grateful to all my professors from Manipal University for preparing me with all the prerequisites necessary for the study program at Eindhoven. Finally, i would like to thank my friends, Sadashiva Prabhu, Suhas Pai, Shrey Patel, Sushma Prakash, and Amulya for their support and guidance.

Dedicated to my parents, family and my beloved brother

Chapter 1

Introduction

Today, mainframe computers [1] play an important role in the everyday operations of most world's largest corporations, including many Fortune 1000 companies. A mainframe is the central data repository, or hub, in a corporation's data, processing center, connected to users through less powerful devices such as workstations or terminals. The mainframe often implies a centralized form of computing, as opposed to a distributed form of computing. The general definition of mainframe is "*mainframe is what businesses use to host the commercial databases, transaction servers, and applications that require greater degree of security and availability than is commonly found on smaller-scale machines*" [1]. Mainframes occupy a dominant place in today's e-business environment. In banking, finance, health care, insurance, and government and in many other public and private enterprises, the mainframe computers continue to form the foundation of modern business. The popularity of mainframe computers is mainly due to their reliability, stability, and the result of continuous technical advancement since the introduction of the IBM System/360, the first mainframe computer in 1964.

Since 1960, mainframe computers have steadily grown to achieve enormous processing capabilities. The new mainframe has an ability to serve tens of thousands of end users, manage petabytes of data, and reconfigure hardware and software resources to accommodate changes in workload on the fly while remaining operational, all from a single point of control. Within IBM, the term mainframe is no longer used, instead the term "zSeries" is used, which refer to zero downtime.

Mainframes are often used by IT organizations to host the most important, mission-critical applications. These applications typically include customer order processing, financial transactions, production and inventory control, payroll, as well as many other types of work. Many of today's busiest websites store their production databases on a mainframe host. Mainframe hardware and software products are ideal for web transactions because they are designed to allow huge numbers of users and applications rapidly

and simultaneously access the same data without interfering with each other. Thus security, scalability, and reliability are critical to the efficient and secure operation of contemporary information processing.

1.1 Context

For corporations, one of the most valuable resources is its data. This critical data should be securely managed and controlled, and simultaneously made available to other users who are authorized to use it. In an IT environment, data security is defined as protection against unauthorized access, transfer, modification, or destruction. Over time, it has become easier for one to create and access to digital information. No longer the system access is limited to system programmers. More and more people are becoming increasingly dependent on the computer systems and the information stored in them. As the number of people using the computer systems increases, the need for data security has become highly important.

In the world of IT security, identity theft remains the top cyber crime with over $56 billion lost to identity theft in 2009 [36]. With the continued growth in credit card use and electronic data transactions, the number of data breaches and personal information lost to cyber crime continues to increase at an alarming rate. The most recent study by Verizon Business shows a steady increase in percentage of data breaches perpetrated by company insiders, with substantial growth in this activity by organized crime data from within the enterprise as shown in Figure 1.1 [36].

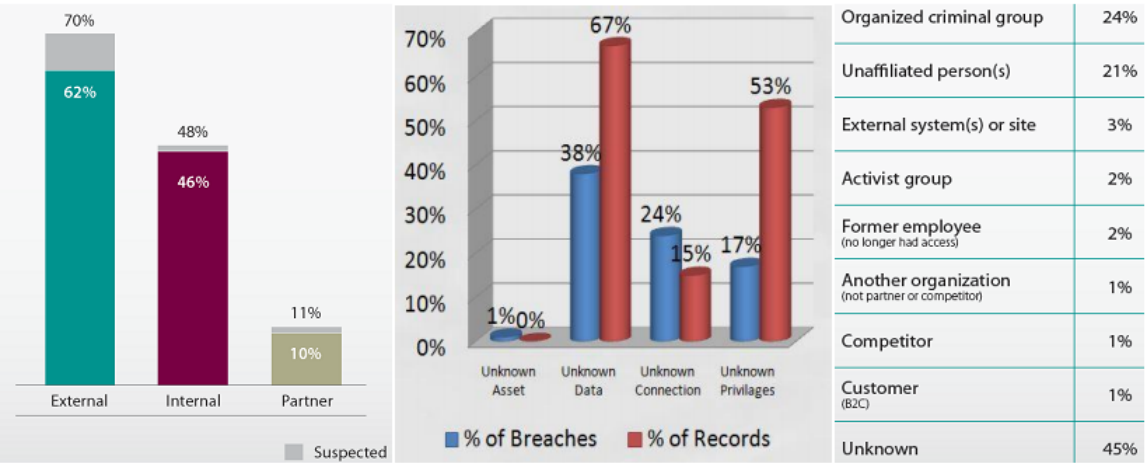


Figure 1.1: Statistics of identity theft and data breach

This study also highlights a fact that almost 70 percent of records lost to cyber crime were records that were unknown to the enterprise. However, the most concerning fact

from this report was that almost half of insiders and external entities responsible for data breaches did not act alone. This clearly shows a growing trend that organized crime is now more likely to gain access to confidential data from within the enterprise [36].

Since mainframe handles over 70% of business critical, transactional, and customer data, security within the enterprise become a matter of high concern [36]. The huge volume of data that is processed and stored, as well as unique storage methodologies used for over 40 years, present daunting challenges to any corporation looking to locate and protect all sensitive data stored in mainframe environments. Mainframes are typically massive and complex environments where manual discovery is impossible and automated solutions have become more demanding to avoid the above said problems.

In order to improve the security of a system, standardization groups are promoting good security practices, regulations and guidelines. These generally include criteria and several rules for hardening a system against the most common forms of attacks. The rules deal with issues related to the transparency and accuracy of financial records, the retention of records within the corporation, and requirements of disaster recovery, business continuity, personal information handling, actions required in the event of data breach, corporation's critical infrastructure, user identification, and access control. *The process that the organization operates in accordance with these predefined guidelines is called Compliance management and the fact that they have or require a procedure is called governance.* Failure to comply with the regulations can cause serious damage to the security of a system and have financial consequences as well. This includes loss of confidential and sensitive data, improper access control that gives rise to data breaches. Financial consequences include huge amount of penalties. Penalties resulting from lack of compliance with major regulations can be stiff. In US, the Sarbanes Oxley Act (SOX) calls for penalties of up to 20 years of jail for altering corporate records, while Healthcare Insurance Portability and Accountability Act (HIPAA) calls for penalties of up to $20,000 for health care providers who fail to maintain the privacy of patient information [24]. These financial issues are one of the factors in making compliance management a top concern in most enterprises.

Compliance management generally consists of various processes such as analysis of regulations, control extraction, classification of controls, and compliance checking etc., in order to achieve and maintain compliance in the system. Compliance checking is one of the important processes in compliance management which can be defined as *the process of verifying whether or not a given system fulfils the compliance requirements of a regulation.* As the complexity of IT systems increases, the complexity and the scope of compliance checking also increases significantly. Compliance checking can be done either manually or in an automated way. However, manual compliance checking is often error prone, the cost is significant, and requires huge labour and time. For example, Financial Executives International (FEI), a leading Chief Financial Officers (CFO) organization, reported that their companies' first-year SOX compliance costs averaged $4.36 million, which was 39 percent higher than they had expected those costs to be in an earlier

survey [24]. Unless automated, a massive compliance effort such as this may affect the revenues and corporate financial performance. Companies need to be able to easily cope up with the frequent changes in regulations. Therefore, Manual compliance is simply not a long term solution as either the regulations are subjected to change very often or the number of regulations increases over time. Automated compliance checking is preferred over manual compliance checking as it gives more accurate results, cost effective, and consumes less time to check the compliance of a system. Automated compliance systems utilize computer systems to perform compliance testing. To achieve compliance automation, automation of auditing and reporting is essential.

The mainframe is a critical element in any IT compliance initiative. It often houses some of the enterprise's most critical IT assets, both data and applications, and therefore must have strong and effective controls over use of these assets. Since, mainframe environments are too complex, manual compliance checking requires huge amounts of labour, cost, and time. In addition, manual compliance checking for such environments may not provide accurate results. As a result, organizations prefer to have an automated compliance checking process to assess compliance of a mainframe system.

Without a consistent and auditable set of controls across all major system platforms, an organization will not be able to achieve and maintain compliance in a cost-effective manner. In order to address the problems of manual compliance checking, IBM is developing an automated compliance checking tool for z/OS and this thesis is a result of the same project.

1.2 Objectives and Contribution

During the course of the thesis, the following research question was investigated.

"How to automate the process of compliance checking of z/OS against various regulations across multiple domains"

The approach taken to answer the research question is as follows:

- Analyse the regulations to determine various factors such as resources and domains to which the rule applies, different classes of data and people specified in regulations, and compliance requirements.
- Classify security controls based on the extent to which they can be automated for the underlying system.
- Define a methodology to implement an automated compliance management pro-

cess.

- Implement an assessment routine using the proposed methodology.

The main challenges that industries have been facing is to come up with techniques and tools to efficiently deal with compliance management. Efficiency may be in terms of the cost, labour costs, time needed to test compliance. Many methodologies [17], [28], [16] have been proposed in the past to implement an efficient automated compliance checking solution. However, these methodologies fail to address the following problems as discussed in Chapter 2.

- Multiple Regulation Types: Often, organizations are compelled to comply with multiple types of regulations. The approach must be able to support multiple types of regulations in compliance checking.

- Flexibility: Often, regulations are subjected to change. The security controls in the regulations are modified to further improve the security of a system. The compliance checking tool must be flexible enough to easily cope up with these changes.

- Multiple domains: In general, regulations include several rules that apply to various domains. Hence an efficient compliance checking tool must be able to check compliance of a system across multiple domains rather than a specific domain.

- Extensibility: As discussed earlier, regulations are subjected to frequent changes. New rules/controls will be added to the regulations to further improve the security of a system. Hence, an efficient compliance checking tool should be extensible in terms of the number of controls. Re-usability of the code helps to easily add new rules to the routine thus making the tool extensible.

- Complexity: Complexity here refers to the complexity of modelling a rule or the control. A simple compliance checking language can be used to formulate the rules rather than complex languages.

- Compliance cost and time: The main problems associated with compliance checking is significant cost and time required to check the compliance of a system. An efficient compliance checking process must consume less time for checking the compliance of a system and should be cost effective.

- Compliance progress traceability: Compliance management is a continuous process. In every iteration of compliance management, the organization tries to improve upon the compliance status of a system. Hence, the compliance report or compliance results generated in every iteration should be able to present the compliance progress achieved in the current iteration. This helps in tracking the compliance progress of every iteration.

Apart from the research question, this thesis also aims to address the above mentioned shortcomings facing by current automated compliance management approaches.

This thesis gives a brief introduction to z/OS, a mainframe operating system and IBM Security zSecure suite. It also gives a brief introduction to the External Security Manager by IBM, RACF that facilitates efficient access control mechanisms in z/OS. It also enables logging and reporting on unauthorized attempts at gaining access to the system and to the protected resources. A brief introduction to a compliance checking language, known as Common Auditing and Reporting Language (CARLa) is also given in this thesis. CARLa is used as a compliance checking language in the implementation of automated compliance checking process for z/OS in this thesis.

The various issues associated with current automated compliance management approaches such as complexity of the compliance checking language, lack of flexibility to cope with changes in regulations, inability to support multiple types of regulations, inability to check compliance across multiple domains, huge amount of cost and time required to check compliance of a system have been identified and addressed in this thesis. In addition, consequences of non compliance are also presented. This thesis also presents an event based classification technique to determine the percentage of automation of the security controls of each regulation for the underlying system. Finally, a compliance checking routine has been implemented to asses z/OS system for compliance with various regulations across multiple domains.

The rest of the thesis is structured as follows:
Chapter 2 presents the related work which summarizes the already existing tools, methodologies of compliance management, and their limitations. Chapter 3 gives a brief description of z/OS, mainframe operating system, the External Security Manager used by z/OS. The chapter also introduces IBM zSecure Audit that is used in automated compliance management of z/OS. The objective of this chapter is to give an abstract idea of mainframe operating system, z/OS, zSecure Audit, and RACF. Chapter 4 proposes a general methodology for implementing automated compliance checking. Chapter 5 presents the implementation of assessment routine for automated compliance checking of z/OS. In addition, a brief introduction to CARLa (Auditing and Reporting Language) is given. Chapter 5 also discusses how the Assessment routine presented in the thesis address the limitations facing by current approaches. Finally, Chapter 6 presents the conclusions, summarizes the contributions, and discusses future research directions.

Chapter 2

Related Work

The demand and the practice for the organizational compliance with regulations have existed for a long time, since security became matter of higher concern. The advent of new technologies and computers since late 20's has been very advantageous in the automation of compliance checking process and has proven helpful in alleviating the problems of manual compliance checking by providing options like auditing secure information which plays an important role in automation of compliance checking and with the invent of auditing and reporting languages. In the remainder of the chapter, we review the scientific literature on automated compliance checking as well as existing industry tools.

2.1 Literature Survey

In the recent years, there has been a lot of work in the area of automated compliance checking.

Y. Liu et al introduced OPAL, a compliance checking framework and related tools, including static method to check business process models against compliance. In this, the overall compliance checking process involves 6 major steps, step 1 is modeling business processes using BPEL (Business Process Execution Language), step 2 involves specifying compliance rules using visual BPSL (Business Property Specification Language), step 3 is further divided into 2 steps, 3.1 aims at transforming the BPEL process model into a representation using pi-calculus, whereas in step 3.2, the pi-calculus is transformed into a finite state machine (FSM). In step 4, the BPSL compliance rules are transformed into linear temporal logic (LTL). In step 5, model checking technology is used to statically verify whether the business processes comply with the imposed regulations. Finally in step 6, counter examples (i.e., execution orders of process activities that demonstrate

how the compliance rules have been violated) are fed back to the business process layer to demonstrate how the compliance rules have been evaluated.

A framework has been introduced in [16] that help health information custodians analyse and improve their business processes in order to comply with relevant privacy legislation. Software requirement engineering technology, called Users Requirement Notation (URN) was used to model the goals and business processes related to the access of confidential information stored in a data warehouse and then the compliance is checked using compliance links. However, the system has a domain specific framework and is hard to update, reuse and modify. Also, the framework fails to support multiple types of regulations in compliance management. The proposed framework allows compliance checking against only HIPAA.

In Reference [17], a compliance management framework was developed that is flexible with regard to change in both business processes and regulations. The paper discusses about eight requirements needed for a compliance management framework. The requirements are: Complexity, Efficiency, Cost, Enforceability, Scalability, Impact Analysis, Change management, and Traceability and Accountability. The paper also presents the various types of compliance checking: Design-time/run-time compliance checking, Forward/Backward compliance checking, and Active/passive compliance checking.

An exhaustive analysis of requirements of compliance checking was carried out in [18], where the authors proposed a policy based framework for semantic business process compliance management. The goal of the paper was to showcase how using semantics, policy management and rule management can automate compliance checking. An architecture has been presented in the paper in order to illustrate their approach. However, the framework requires policies to be transformed into sets of business rules. It requires the definition of set of transformations. A first transformation is needed to generate business rule models out of the policy definitions. A second transformation is needed to generate operational rule models out of business rule models. In addition, the framework is based on the policy. Hence, it does not support other forms of regulations such as security standards, security guidelines, and best practices.

An ontology and semantic rule based approach that facilitates the compliance management checking was proposed in [19]. Exception handling approach was used to create a robust rule base model that deals with data incompleteness, one of the major deficiencies in the semantic web. The authors tried to semi-automatically extract compliance knowledge from ISO17799. They have used Semantic Web Rules (SWRL) to represent the compliance requirements of the controls. However, the complexity of representing the compliance requirements using SWRL is high.

A methodology was proposed in [21] which focuses on two main objectives: (1) to alleviate the current problems by providing increased adaptability and re-usability, (2) to enhance automation towards extracting regulatory information and modeling them into

a deep semantic ontology. The proposed methodology consisted of four major phases: 1) Extraction of the regulations, 2) Formalization of the extracted regulations, 3) Mapping the regulations to the validation tasks, 4) Generation of semantic rules. Two more additional steps were added as enhancement, 5) Communication of the findings to the user, 6) Monitoring of changes in the system. However, the scope of the compliance management was limited to a single domain, i.e. pharmaceutical industry and takes only one regulation, Endralex European regulation [22] into consideration. Also, the implementation was based on already existed ontology OntoReg [23] ontology for the pharmaceutical industry.

Marwane El Kharbili et al presents the definitions of regulatory management, compliance checking, and compliance modelling. They proposed with a compliance representation language (COREL), which is designed for compliance modelling. It allows modular modelling of compliance decision-making that achieves flexible compliance. COREL is policy based compliance checking language that use policies as the core concepts for modeling compliance requirements. The various shortcomings that current compliance management approaches facing have been addressed in the paper. COREL supports flexibility, multiple regulation types, and traceability. However, COREL is a Domain Specific Modeling Language. So, it does not support multiple domains in compliance checking. Also, COREL addresses only a single family of security controls i.e., Access Control. Hence, it fails to check the compliance against multiple types of security control families. COREL fail to track the compliance progress of a system.

2.2 Product Survey

Following paragraphs discusses about various compliance management tools that are already existing.

IBM Tivoli Compliance Manager [20] is a computer assisted compliance management process which helps in establishing policies for reporting processes and to provide an organized approach to gathering information about business controls. However, the product only addresses a small subset of problems within compliance management and focus on lower-level aspects of IT governance such as asset management, change management and licensing management. The tool also fails to track compliance progress of a system and time needed to check compliance is significant.

RSA Archer compliance management [30] provides a centralized, access-controlled environment for automating compliance processes, assessing deficiencies, and managing remediation efforts. This is web based software which enables organization to document process and technical controls, links them to authoritative sources, performs risk-based scoping, execute design and operating tests, and responds to gaps. It also enables reporting assessment results and remediation activities to management and regulators through

real-time dashboards. However, it fails to cope up with the addition of new regulations. The customised software allows checking compliance against subset of Sarbanes-Oxley, sections 302 and 404, J-SOX, Gramm-Leach-Bliley, and other requirements. In addition, the software cannot track compliance progress of a system.

ComplianceManager Tool [31] by Compliance Manager [1], provides a common framework and an integrated approach to manage all compliance requirements faced by an organization. It enables companies to manage cross-industry mandates and regulations through risk management, document control, compliance training, continuous auditing, as well as recording and reporting of exception events and required remediation. The solution helps identify, test and remediate key compliance controls via self-assessments. It allows compliance checking against wide range of regulations such as SOX, Turnbull Guidance, Bill 198, and OSHA etc. However, the tool fails to keep track of the compliance progress of a system.

Security Compliance Manager [32] by Microsoft [2] enables one to quickly configure and manage both desktops and servers using Group Policy and Microsoft System Center Configuration Manager. However, the tool supports compliance management based on only two regulations, Industry best practices and Microsoft security guide recommendations. Also, the tool cannot be used for other environments such as mainframes. The tool also fails to keep track of compliance progress of a system.

Accelus Compliance Manager [33] by Thomson Reuters [3], is a compliance management solution that is capable of coping up to the increase in the regulations. The system is designed to enable users to assess and manage regulatory events, connect regulatory change to relevant policies and controls, monitor, test, and audit controls, and provide a central system to evidence and report compliance to management and regulators. However, the tool fails to keep track of compliance progress.

Compliance Manager for z/OS [35] by CA [4], is designed to automate compliance management tasks that are currently being done manually, and making it easier for mainframe administrators to monitor and track changes that might impact regulatory compliance or security. The key features of this tool are:

- Stores information about mainframe security events in a relational repository that is accessible for compliance reporting, allowing complex reporting processes to be initiated.

- Detects and records changes to CA ACF2, CA Top Secret, and IBM RACF configurations, operating system security configuration, and selected PDS/PDSE data

[1] http://www.mycompliancemanager.com/
[2] http://www.microsoft.com/
[3] http://www.accelus.thomsonreuters.com/
[4] http://www.ca.com/

sets identifying and recording specifically what was changed.

- Provides real-time notification of potential security breaches indicated by changes in the security configuration and specific security events.
- Writes information about mainframe security events to a dedicated z/OS log stream.
- Web enabled user interface for compliance reporting.

However, the time taken for checking the compliance of a z/OS system is still significant and the tool also fails to track compliance progress of a system.

Chapter 3

Introduction to z/OS, zSecure, and its External Security Manager

In this chapter, a brief introduction to z/OS, a mainframe operating system is given. The chapter also gives a brief description of External Security Manager (ESM) of z/OS. The rest of the chapter is structured as follows: Section 3.1 discusses about the basics of z/OS and discusses about the hardware resources used by z/OS. Section 3.1.1 describes the term dataset in z/OS and discusses the types of datasets in z/OS. In Section 3.1.2, the external security manager RACF is introduced and also explains how it provides security in z/OS. Section 3.1.2 discusses about logging and reporting which plays an important role in compliance checking process. Finally, Section 3.2 gives a brief introduction to the zSecure Audit that provides solutions for automated auditing and compliance reporting on mainframes.

3.1 z/OS

An operating system is a collection of programs that manage the internal workings of a computer system. z/OS [2] is a widely used mainframe operating system, it has been designed to offer a stable, secure, and continuously available environment for applications running on the mainframe. z/OS evolved from an operating system that could process a single program at a time to an operating system that can handle many thousands of programs and interactive users concurrently.

In most early operating systems, requests for work entered the system one at a time.

The operating system processed each request or job as a unit, and did not start the next job until the one being processes had completed. This arrangement worked well when a job could execute continuously from start to completion. However, often a job had to wait for information to be read in from, or written out to, a device such as a tape drive or printer.

Input and Output (I/O) take a long time compared to the electronic speed of the processor. When a job waited for I/O, the processor was idle. Finding a way to keep the processor working while a job waited would increase the total amount of work the processor could do without requiring additional hardware. z/OS addresses this issue by dividing it into pieces and giving portions of the job to various system components and subsystems that function interdependently. At any point in time, one component or another gets control of the processor, makes its contribution, and then passes control along to a user program or another component.

The z/OS operating system executes in a processor and resides in processor storage during execution. Mainframe hardware resources used by z/OS consist of processors and a multitude of peripheral devices such as disk drives (called direct access storage devices (DASD)), magnetic tape drives, and various types of user consoles. Figure 3.1 gives a pictorial representation of hardware resources used by z/OS.

We can interact with z/OS in mainly two modes: batch mode and interactive mode. Batch mode allows batch processing that is running of jobs on the mainframe without user interaction [2]. A batch job is submitted on the computer, reads and processes data in bulk and produces the output. Interactive mode allows users to interact directly with the system using commands and menu style user interfaces. z/OS provides a number of facilities to allow users to interact directly with the operating system.

Time Sharing Option (TSO): TSO provides a single user logon capability and basic command prompt interface to z/OS. Figure 3.2 shows a TSO logon screen in z/OS. Interactive System Productivity Facility (ISPF): ISPF is a menu driven interface that provides a collection of menus and panels which offers a wide range of functions to assist users in working with data files on the system. Figure 3.3 shows ISPF menu interface, which is used to interact with z/OS. z/OS UNIX interactive interfaces: This facility allows users to write and invoke shell scripts and utilities, and use shell programming language.

3.1.1 Datasets in z/OS

In z/OS, a *dataset* refers to a file that contains one or more records, where a *record* is a fixed number of bytes of data. Datasets are generally used to store information needed by applications or the operating system such as source programs, macro libraries, or system variables or parameters. The term *field* is a component of a database which can

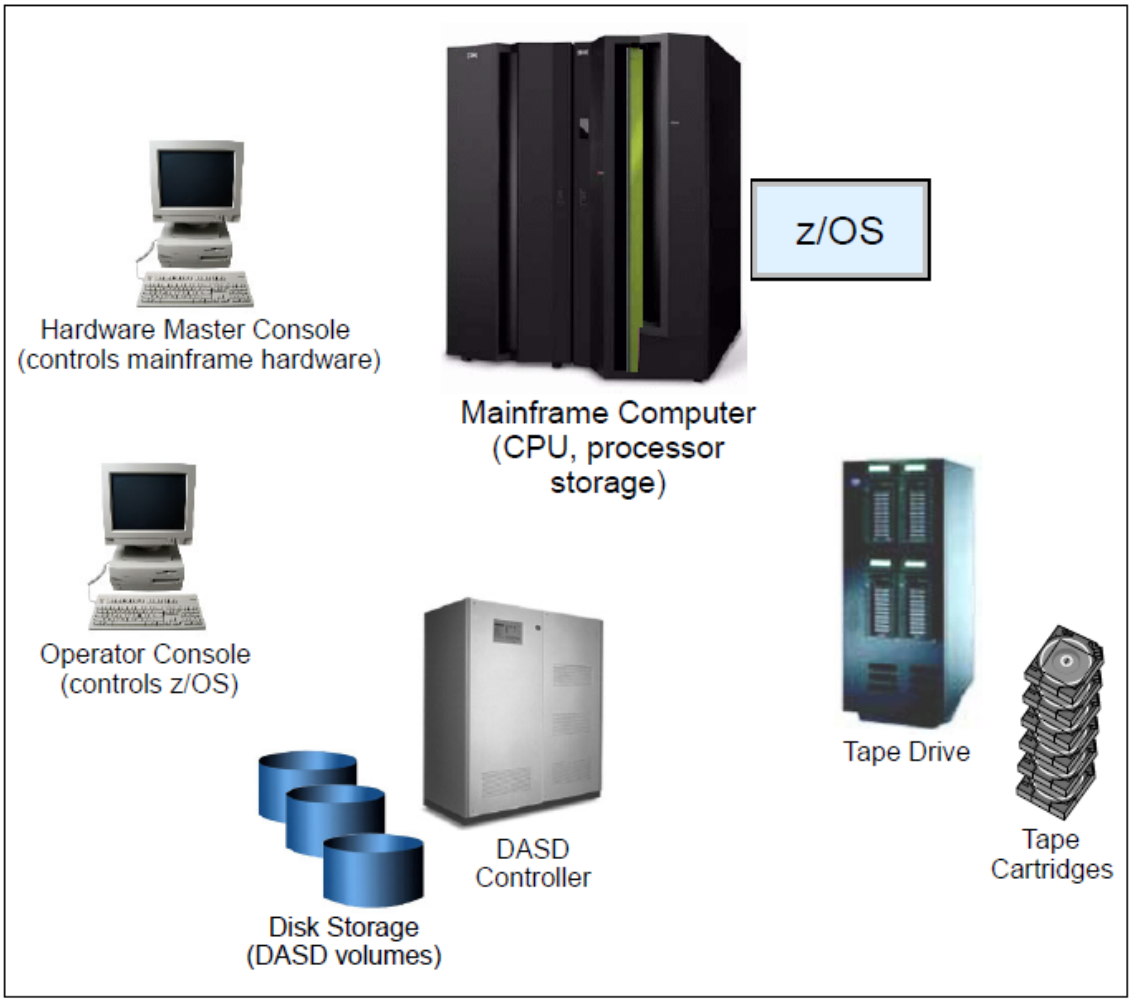


Figure 3.1: Resources used by z/OS

store character, numeric, flag, or hexadecimal data. The field names can be used in the program once they are defined.

There are three types of dataset in z/OS: sequential, partitioned, and VSAM datasets. In a *sequential dataset*, records are data items that are stored consecutively.
A *partitioned dataset* (PDS) consists of a *directory* and *members*. The directory holds the address of each member and thus makes it possible for programs or the operating system to access each member directly. Each member, however, consists of sequentially stored records or programs. A partitioned dataset is commonly referred to as a library. In z/OS, libraries are used for storing source programs, system and application control parameters, JCL, and executable modules. Job Control Language (JCL) is used to tell the system what program to execute, followed by a description of program inputs and

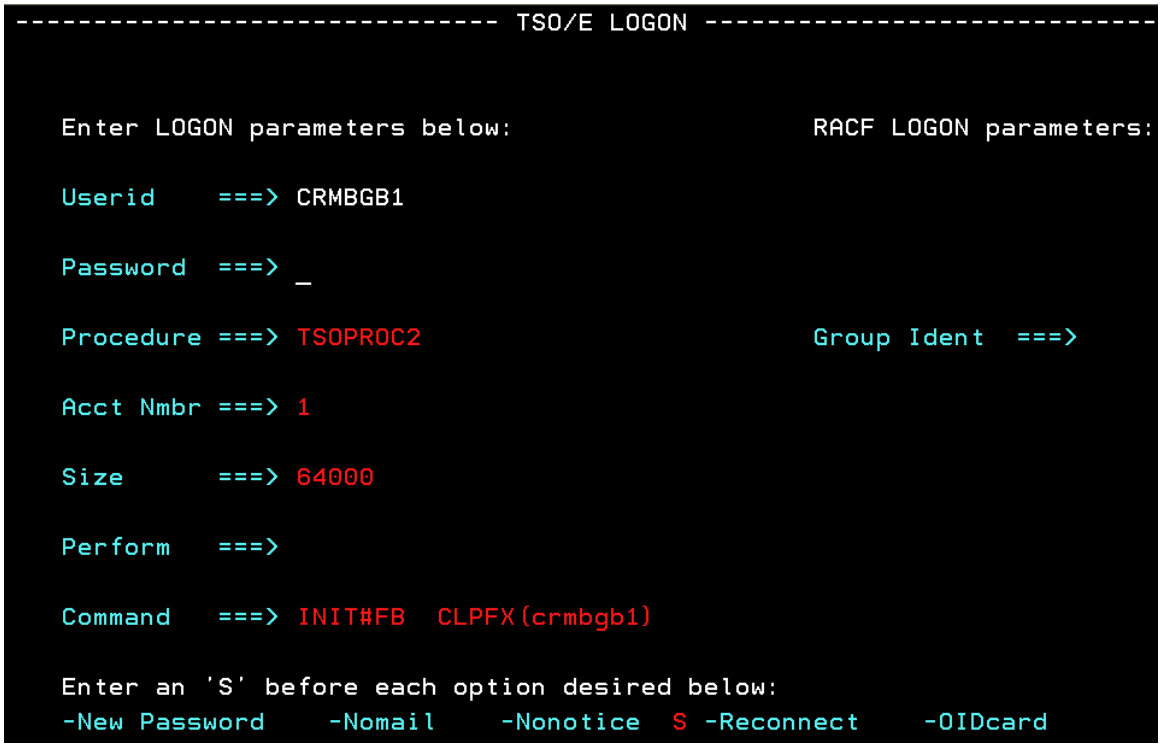


Figure 3.2: TSO logon screen

outputs. It is possible to submit JCL for batch processing or start a JCL procedure (PROC), which is considered a started task.

In a Virtual Storage Access Method (VSAM) key sequenced dataset (KSDS), records are data items that are stored with control information (keys) so that the system can retrieve an item without searching all preceding items in the data set.

3.1.2 Security in z/OS

Mainframe based systems implement security using External Security Manager (ESM). The ESMs used by z/OS are Resources Access Control Facility (RACF) by International Business Machines (IBM) and Access Control Facility (ACF2), Top Secret (TSS) both by Computer Associates (CA). Since, compliance checking is done using records logged by RACF in the thesis, it is explained briefly in the following paragraphs.

RACF provides tools to manage user access to critical resources. RACF is an add-on software product that provides basic security for a mainframe system. RACF protects resources by granting access only to authorized users of the protected resources. RACF retains information about users, resources, and access authorities in special structures

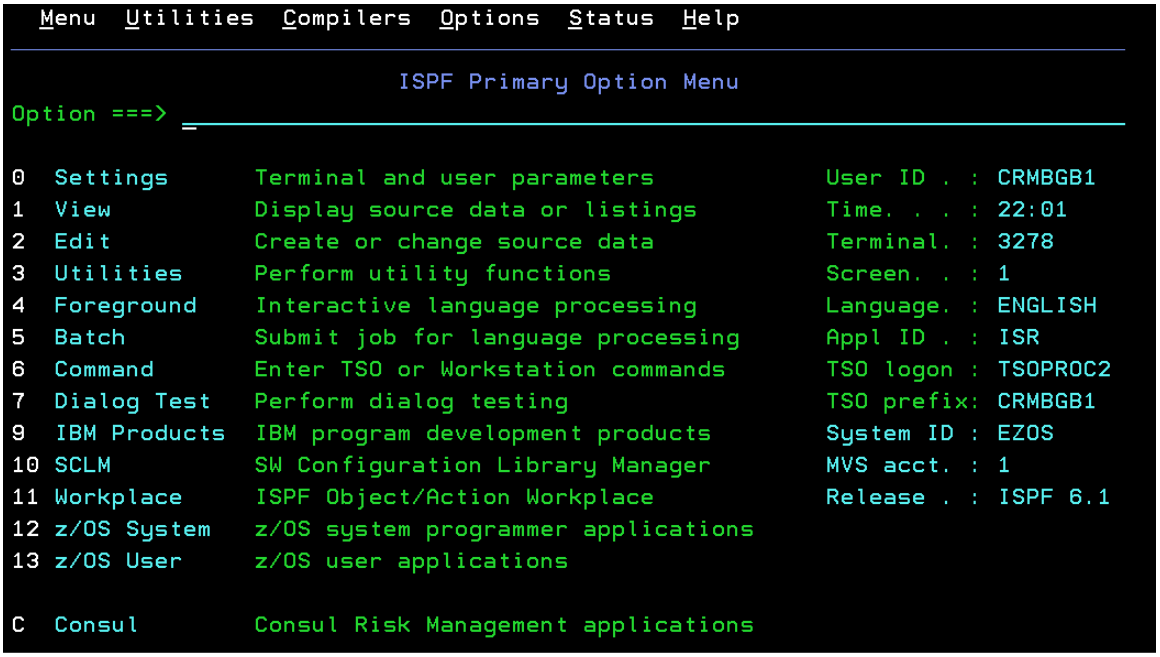


Figure 3.3: ISPF main menu

called profiles in its database, and it refers to these profiles when deciding to grant or allow access to a system resource.

RACF provides the following features in order to establish security in mainframe system:

- Identify and authenticate users
- Authorize users to access protected resources
- Log and report various attempts of unauthorized access to protected resources
- Control the means of access to resources
- Allow applications to use the RACF macros

RACF can also provide security to transactions and files of subsystems such as Customer Information Control Systems (CICS) and Database Manager (DB2).

Logging and Reporting

Logging is a task of writing a history of actions and changes. It is the process of recording of data about specific events which plays an important role on automated compliance

checking process. An *event* can be defined as an occurrence at any point that changes the state of a system. As RACF can identify and verify user's user ID and recognize the resources that can be accessed by the user, RACF can record the events whenever user-resource interaction happens. This function records the actual activities and violations in the system. RACF enables logging by storing statistical information, such as the date, time, and number of times that a user enters a system and the number of times a specific resource was accessed by any one user. RACF also writes security log records when it detects:

- Unauthorized attempts to enter the system
- Authorized or unauthorized attempts to access RACF-protected resources
- Authorized or unauthorized attempts to enter RACF commands
- Modifications to profiles on the RACF database.

RACF writes records to System Management Facility (SMF) for detected, unauthorized attempts to enter the system. RACF allows listing the contents of these records. These records can then be used to detect possible security exposures or threats. In addition, these records can also be used while checking compliance of a system to regulations. RACF can monitor the system along with other applications and subsystems of z/OS. They are: Job Entry Subsystems (JES), Time Sharing Option (TSO), Information Management Systems(IMS), Customer Information Control Systems (CICS), Storage Management Subsystem (SMS), Database Manager (DB2), VTAM, Advanced Program to Program Communication (APPC) sessions, Terminals etc.

JES is a subsystem used by z/OS to receive jobs into the operating system, to schedule them for processing by z/OS, and to control their output processing. In addition, JES also provides supplementary job management, data management, and task management functions, such as scheduling, control of job flow, and the reading and writing of input and output streams on auxiliary storage devices, concurrently with job execution.

CICS is a general purpose transaction processing subsystem of z/OS. CICS provides services for running an application online by request, while there are several other users submitting requests to run the same applications, using the same files and programs.

In z/OS, IMS is both a transaction manager and database manager. IMS consists of three other components: the Transaction Manager (TM), Database Manager (DB), and a set of system services that provide common services to the other two components.

3.2 Introduction to zSecure Audit

In this section, the components of the IBM zSecure Suite that are used in relevance with automated compliance checking are introduced. Since, zSecure Audit is the only component used in compliance management, it is described at a high level. However, IBM zSecure Suite consists of other components too.

3.2.1 zSecure suite

The IBM zSecure Suite [7] consists of multiple modular components and tools, which are designed to help manage RACF database quickly and efficiently. The components of zSecure are: IBM zSecure Admin, IBM zSecure Audit, IBM zSecure Visual, IBM zSecure CICS Toolkit, IBM zSecure Alert, and IBM zSecure command verifier. zSecure can help in monitoring for threats, conduct status audits, help manage control self-assessments, and assist with the enforcement of policy compliance. zSecure consists of CARLA program which is used as the compliance checking language in this thesis. Figure 3.4 depicts all the components available in IBM zSecure suite. zSecure suite is categorized into two domains:

- Security Audit and compliance
- Administration management

The operating systems supported by zSecure Suite are z/OS and z/VM. A brief introduction to z/OS is given in Section 3.1. The External Security managers supported by IBM zSecure are:

- IBM RACF
- CA ACF2
- CA Top Secret

In this thesis, we focus only on RACF events in compliance checking of z/OS.

3.2.2 zSecure Audit

zSecure Audit [7] is a comprehensive mainframe compliance and auditing solution. It enables analysis and reporting on mainframe events and automatically detects security exposures and misconfigurations with the help of status auditing and automated analysis using a built-in knowledge base. zSecure Audit has extensive change tracking facilities,

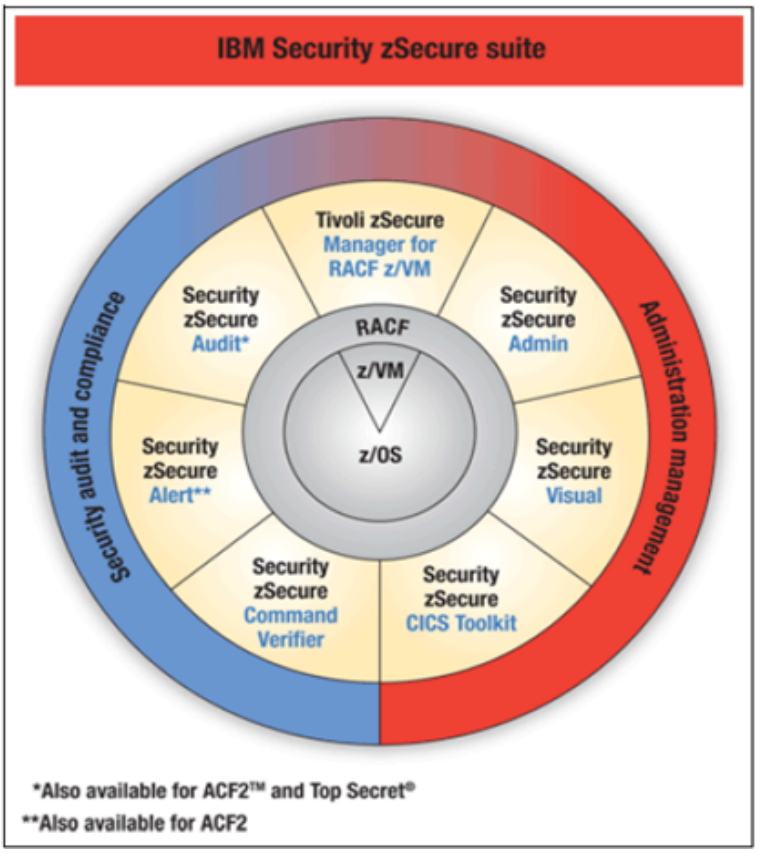


Figure 3.4: The IBM zSecure suite

which assists in establishing a security baseline and automatically track changes to it. zSecure Audit is generally used by IT auditors and security personnel to help meet compliance and audit requirements. Security personnel use the software to:

- Interrogate audit logs, such as SMF, to analyze mainframe events, detect security breaches, perform trend analysis, and fix problems.
- Conduct regular security reviews to asses the current state of system security.
- Set up continuous automated auditing to track changes and highlight exposures.
- Use validation utilities to help maintain a secure and clean RACF database.

IT auditors often refer to zSecure Audit as a Computer Assisted Audit Technique (CAAT) and may use the software as part of an audit of z/OS or business application(s) to help them assess security controls. An IT auditor may follow a set of documented audit procedures during an audit to help them in the assessment process. zSecure Audit

can help automate this process through running either predefined or customized reports. The majority of documented z/OS audit procedures in use by IT auditors today are already built into predefined reports within zSecure Audit. System programmers are often benefited from zSecure audit as it helps assess the status of z/OS, such as supervisor calls (SVCs), exits, and program property table (PPT) and authorized program facility (APF) settings to determine where audit concerns exist.

zSecure audit helps in reducing the costs associated with reporting and analysis through reporting and automation capabilities. zSecure Audit can correlate information from several different sources and systems, helping consolidate reporting and significantly strengthen controls. Reporting is available in various formats, including Extensible Markup Language (XML).

3.2.3 zSecure Audit architecture

IBM zSecure audit provides the most comprehensive analysis available for z/OS security posture by correlating data from various input sources. zSecure can correlate data from the following:

- External Security Manager (ESM) security database(s).
- The z/OS IPL parameters and other configuration information from multiple systems.
- The System Management Facility (SMF) audit trail data from multiple systems.
- Other sources (HTTP logs)

The information gathered about system by zSecure audit is stored in a database structure that can be analyzed in a variety of ways using the supplied reports. Customized reports can be added using Common Auditing and Reporting Language (CARLa). zSecure audit can be used in two modes: interactively using an ISPF based set of panels or in a batch mode. Figure 3.5 [7] shows the data flow architecture of zSecure audit. zSecure Audit can also report on ACF2 databases and events and the CA-TSS Audit/Tracking file.

3.2.4 Initial setup of zSecure Audit

Figure 3.6 shows the zSecure main menu after the AU option has been entered in ISPF menu, zSecure audit features are also available using RA (RACF profile reports), EV (events) of the zSecure main ISPF menu. Figure 3.7 shows the list of setup (SE) options in zSecure audit for RACF. The options in SE include Run through which we can specify run options such as selecting permanent work datasets etc., Input files that enables

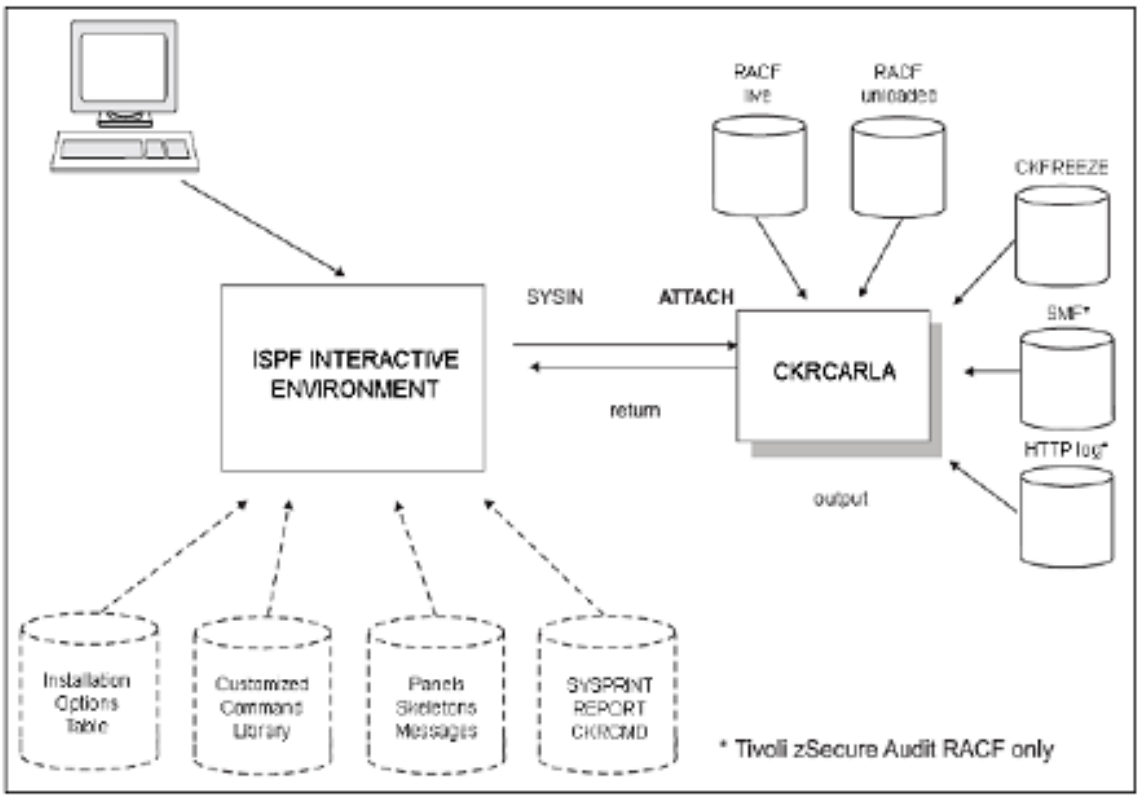


Figure 3.5: zSecure Audit data flow

selecting the input datasets to work with, New files that enables creating a new dataset such as RACF dataset or CKFREEZE dataset, and etc. SE.1 option can be used to select the input data that is used for this zSecure audit session. The input sources must be selected before the user can report on the desired events or environments. Different reports are available based on input files that are selected. The number of input sources available for reporting can be limited based on privileges. Figure 3.8 depicts database selection panel of zSecure Audit after entering SE.1 option in main zSecure Audit main menu. **Note:** only a subset of databases is listed in the above figure as it is limited by the privileges of the user. Input sources supported for RACF are:

- Active RACF primary database
- Active RACF backup database
- Copy of a single dataset RACF database
- Copy of first dataset of a RACF database

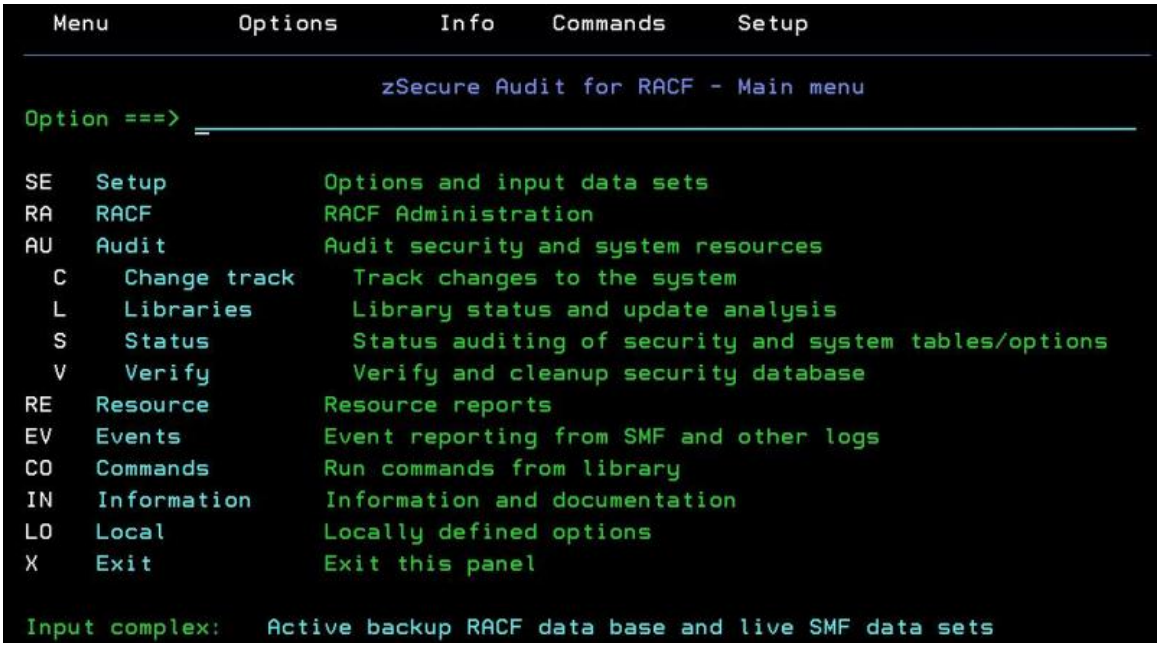


Figure 3.6: Invoking zSecure Audit from main menu

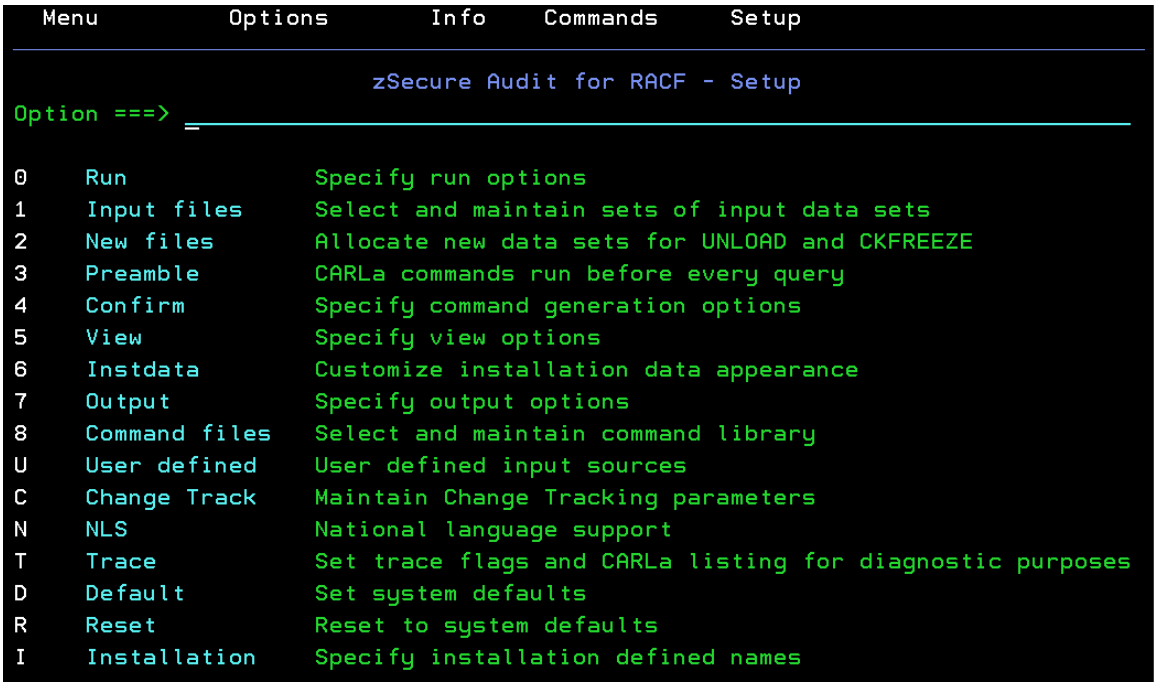


Figure 3.7: List of Setup (SE) options

- Copy of a subsequent datasets of a RACF database

Figure 3.8: Database selection panel in zSecure Audit

- RACF database unloaded by zSecure

The UNLOAD dataset is created by zSecure Audit UNLOAD command and it will never contain RACF passwords, copy of RACF databases are the exact copies of RACF database. These copies can be used as a live RACF database and therefore contain all sensitive fields, such as the user authentication values in encrypted form. zSecure audit can report on events logged to SMF. The SMF data can be live or active SMF, previously dumped to disk or tape, or SMF logstream. zSecure audit provides an extensive analysis of the z/OS operating system configuration parameters that relate to security. The CKFREEZE input files are a point-in-time snapshot of the system settings and additional security related data.

zSecure facilitate automated compliance checking by using zSecure fields. The term *field* is a component of a database which can store character, numeric, flag, or hexadecimal data. The field names can be used in the program once they are defined. zSecure has a set of fields in various report types as shown in Figures 3.9, 3.10, and 3.11.

Figures 3.9, 3.10, and 3.11 shows the subset of zSecure reports types present in zSecure. The figures also depicts other information of each report type such as T2 that represents the code or abbreviation of the report types (For Example: ACCESS has an abbreviation as AM), fields gives the total number of fields defined in each of the report types, Rpt represents the total number of repeated fields where repeated fields are the fields that may contain more than one value, Mod represents the number of modifiable fields in each report type, Subselct represents the number of fields containing the name of one or more subselect groups, and Tag represents the identifier for identifying the report type.

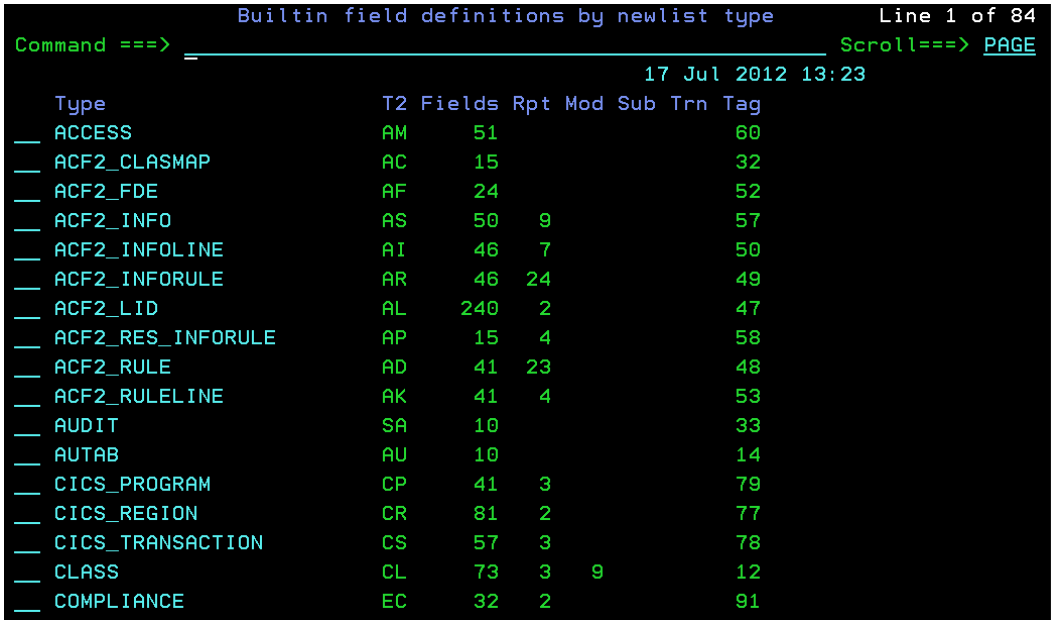
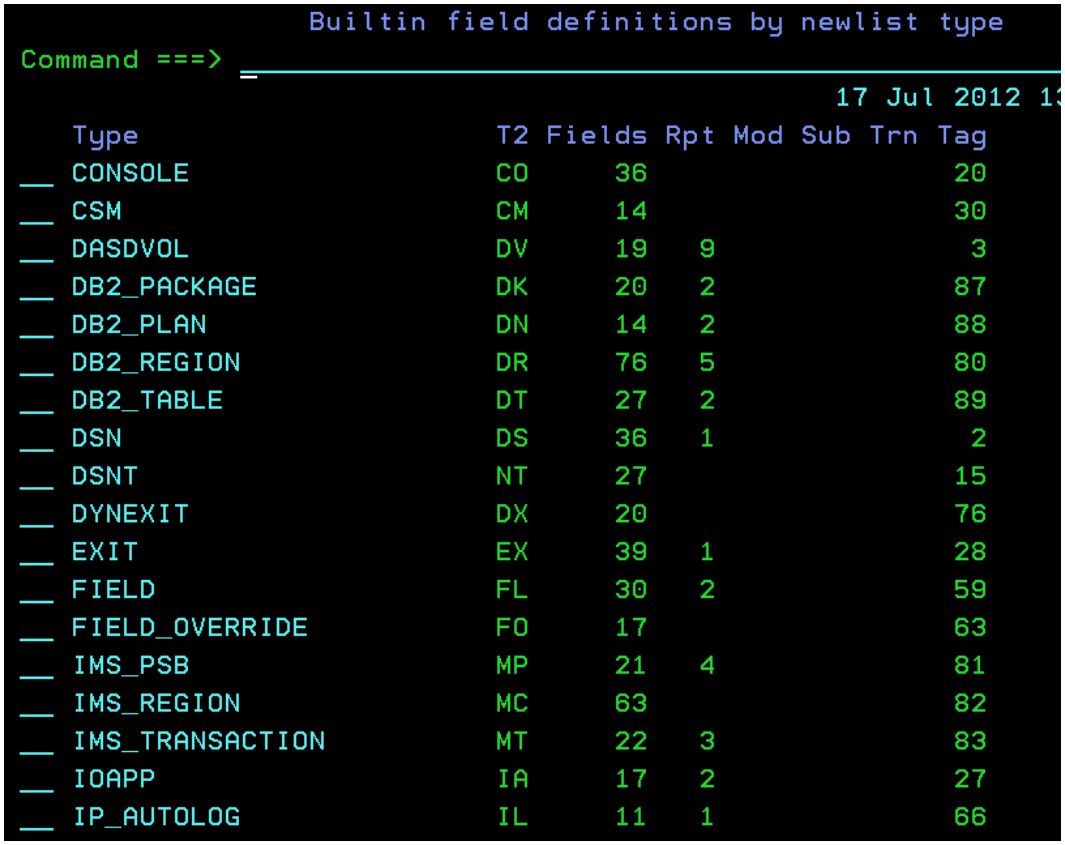


Figure 3.9: List of zSecure Report types-1

Each report type consists of a set of fields defined in it. These fields enables compliance checking of report types of various applications, systems and subsystems of z/OS such as ACF2, CICS Region, and Compliance etc. Figure 3.10 shows a list of some more report types with the total number of fields in each of the report type. Figure 3.11 depicts some more report types which includes Sensitive datasets (SENSDSN), SETROPTS, SMF, and RACF etc. zSecure supports compliance checking on some more report types which are not present in the above figures.



Builtin field definitions by newlist type

Command ===>

17 Jul 2012 1

	Type	T2	Fields	Rpt	Mod	Sub	Trn	Tag
__	CONSOLE	CO	36					20
__	CSM	CM	14					30
__	DASDVOL	DV	19	9				3
__	DB2_PACKAGE	DK	20	2				87
__	DB2_PLAN	DN	14	2				88
__	DB2_REGION	DR	76	5				80
__	DB2_TABLE	DT	27	2				89
__	DSN	DS	36	1				2
__	DSNT	NT	27					15
__	DYNEXIT	DX	20					76
__	EXIT	EX	39	1				28
__	FIELD	FL	30	2				59
__	FIELD_OVERRIDE	FO	17					63
__	IMS_PSB	MP	21	4				81
__	IMS_REGION	MC	63					82
__	IMS_TRANSACTION	MT	22	3				83
__	IOAPP	IA	17	2				27
__	IP_AUTOLOG	IL	11	1				66

Figure 3.10: List of zSecure Report types-2

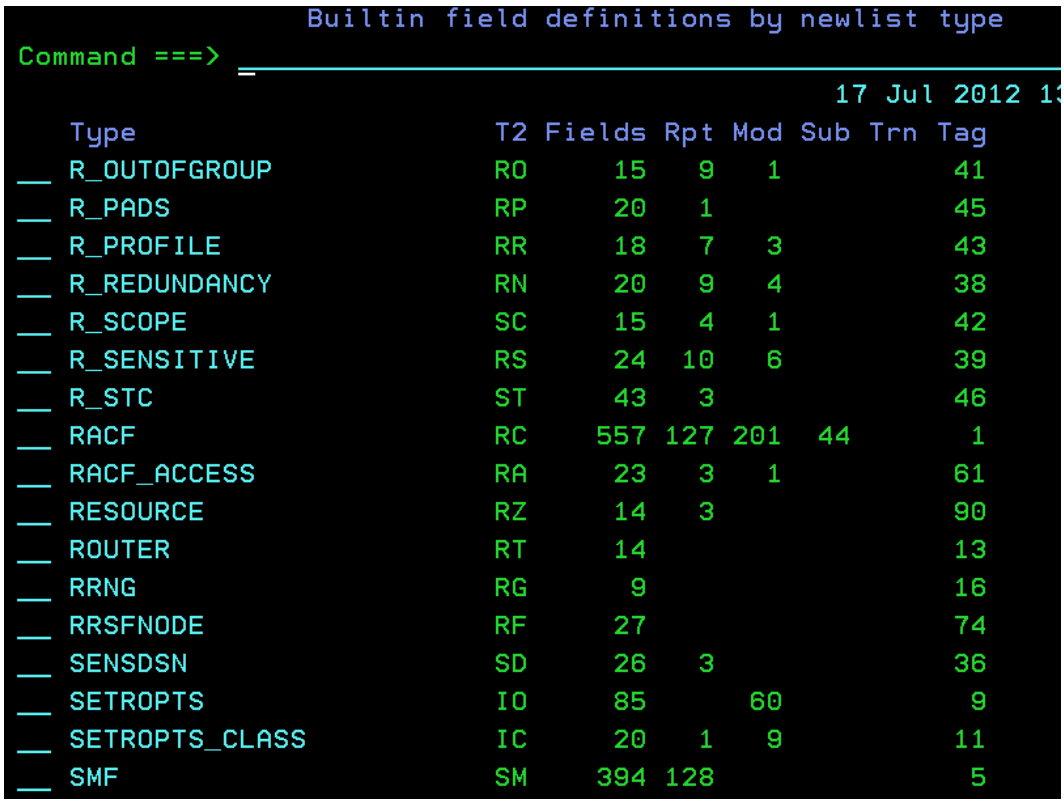


Figure 3.11: List of zSecure Report types-3

Chapter 4

Methodology

Compliance management is the process in which an organization operates according to some regulations. These regulations are often documented in natural language. Each regulation consists of a set of security controls that must be satisfied by the organization to be in compliance with these regulations. *Compliance checking* is defined as the problem of verifying whether or not a given system fulfils the compliance requirements of a regulation. Compliance checking can be done either manually or in an automated way. However, since manual compliance checking has several problems associated with it as discussed in the earlier chapters, automated compliance checking is preferred. In this chapter, a methodology to implement an automated compliance checking process is proposed. This chapter also presents clear definitions of compliance management, regulation, and security controls. It also explains the consequences of non-compliance and advantages of achieving and maintaining compliance in organizations.

The rest of the chapter is structured as follows: Section 4.1 gives a brief introduction to compliance management and discusses the consequences of non-compliance. Section 4.2 proposes a methodology for compliance management which consists of six main processes, Regulation Analysis, Control Extraction and Classification, Test Compliance, Deviation analysis, Deviation handling, and Reporting.

4.1 Compliance Management

Compliance management is the process in which an organization operates in accordance to some regulations. In literature, compliance management is defined as follows:

Definition 1. *Compliance Management is the problem of ensuring that enterprises (data, process, organization, etc.) are structured and behave in accordance with the reg-*

ulations that apply. In the opposite case we say that a company is violating a regulation. [28]

Many organizations are compelled to achieve and maintain compliance to avoid legal consequences and risks of non-compliance. Though consequences of non-compliance vary from regulation to regulation, common consequences include:

- Loss of reputation.
- Loss of customers.
- Loss of market share if competitors show better compliance results.
- Litigation from shareholders.
- Huge amount of penalties.
- Loss of data.
- Affects business goals.
- Strict punishments from judiciary.

These consequences motivates the organizations to comply with the obligatory regulations. However, besides preventing the negative consequences, compliance with regulations also has many advantages. In many organizations, strong compliance can create advantages when compared to competitors and may open up new markets for the products and services. Compliance with privacy regulations such as HIPAA and PCI DSS which are concerned with protecting the patient's records and consumer card information respectively helps to protect confidential/personal data. This helps in building trust with customers and thus improving the market share with them.

4.2 Methodology for Compliance Management

In this section, a methodology for compliance management is proposed and is explained in detail. Achieving and maintaining compliance cannot be done in a single iteration. It is often a repeated/cyclic process. Compliance management consists of several sub processes that must be performed repeatedly in order check and maintain the compliance in a system. Figure 4.1 depicts the proposed methodology which is comprised of six main sub processes, namely Regulation Analysis, Control Extraction and Classification, Test Compliance, Deviation analysis, Deviation handling, and Reporting.

The first task in this methodology is Regulation Analysis. In this process, all regulations

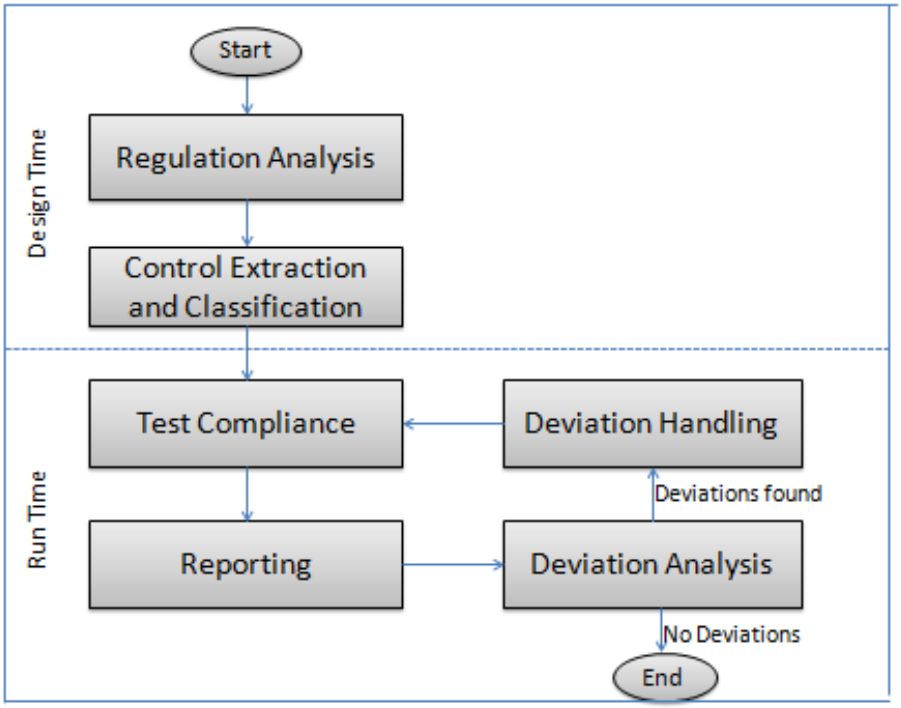


Figure 4.1: Compliance Management

with which a system must comply are analysed to determine various factors such as, type of a regulation, various domains on which rules apply, roles and resources defined in regulations, data classification given in regulations, and security control structure. Once regulation analysis is done, control identification and extraction is carried out in which the security controls that are applicable for the underlying system are identified and extracted from the regulation. After controls have been identified, they are classified into various categories to determine the percentage of automation of the controls for the underlying system. Once controls are classified, they are implemented using a compliance checking language. Then the system is tested for compliance and a compliance report is generated. Compliance report consists of compliance status of all the objects of multiple domains in the system. This is given as input to deviation analysis which is aimed to find the deviations/violations present in the system. This helps to find the source of non-compliance in the system. The deviations that are found in deviation analysis are handled by taking appropriate actions either manually or in an automated way in deviation handling. Now, the system is again checked for compliance and the process continues as earlier. If there are no deviations present in the system, the compliance management iteration reaches the end state.

4.2.1 Regulation Analysis

The purposes of regulation analysis is to analyse regulations to determine various factors such as, type of regulation, roles and resources defined in the regulation, various domains on which regulations apply, and security control structure. Regulation analysis also helps to identify the security controls that are applicable for the underlying system. Examples of the regulations are HIPAA [11], [12], [13], PCI DSS [10], ISO 27002 [14], and DISA STIG [9] etc.

Definition 2. *A Regulation is a document written in natural language containing a set of guidelines specifying constraints and preferences pertaining to the desired structure and behaviour of an enterprise. A regulation specifies the domain elements to which it applies.[27].*

Regulations are categorized into different types. The regulations impacting IT generally fall into one of three major categories:

- **Governance:** These regulations deal with issues related to the transparency and accuracy of financial records, retention of records within the corporation, and requirements of disaster recovery and business continuity. In brief, these regulations are intended to ensure that proper controls exist to guarantee that corporate reporting is accurate, timely, and complete [24]. One of the examples of Governance regulation is SOX [34].

- **Privacy:** These regulations are often specific to a single vertical market, and dictate how a user's personal information must be handled by an organization. There are regulations which also specify what type of personal information may be kept, how the information may be handled, and what actions are required in the event of a breach established privacy restrictions [24]. One of the examples of privacy regulation is PCI DSS [10].

- **Security:** The role of security regulation is to protect a corporation's critical infrastructure. These regulations specify how users will be identified, how their access to sensitive resources must be controlled, and how that access may be tracked and audited [24]. One of the examples of security regulations is HIPAA [11].

Every regulation consists of a set of security rules or security controls. Security controls are generally structured as a set of conditions that must be satisfied to be in compliance with regulation. According to NIST,

Definition 3. *Security controls are defined as the safeguards and countermeasures prescribed for an information system to protect the confidentiality, integrity, and availability of the system and its information [29].*

NIST classifies security controls into eighteen families and associates each security control family with one of three general classes of security controls. Each security control family contains security controls related to the security functionality of the family. A control is associated with a security control class based on the dominant characteristics of the controls in that family. Table 4.1 summarizes the security control families and the associated security control classes as listed by NIST [29]. The security control struc-

Family	**Class**
Access Control	Technical
Awareness and Training	Operational
Audit and Accountability	Technical
Security Assessment and Authorization	Management
Configuration Management	Operational
Contingency Planning	Operational
Identification and Authentication	Technical
Incidence Response	Operational
Maintenance	Operational
Media Protection	Operational
Physical and Environmental Protection	Operational
Planning	Management
Personal Security	Operational
Risk Assessment	Management
System and Services acquisition	Management
System and Communication protection	Technical
System and Information Integrity	Operational
Program Management	Management

Table 4.1: Security control classes and families

ture consists of the following components: a control statement, which provides a concise statement of the security activities needed to protect a particular aspect of an information system, description of the control, list of compliance requirements to be satisfied by the organization or by the information system in order to be in place with the control. However, the control structure differ from regulation to regulation.
The following gives a definition of operational control, a class of security controls.

Definition 4. *Operational controls are defined as the safeguards and countermeasures for an information system that are primarily implemented and executed by people as opposed to systems [29].*

The following example presents an operational control from [29] which belongs to a family of Personnel Security.

Example 1. *Control statement: Position Termination*

Compliance Requirements: The organization, upon termination of individual employment:

- *Terminates information system access.*
- *Conducts exit interviews.*
- *Retrieves all security-related organizational information system-related property and*
- *Retains access to organizational information and information systems formerly controlled by terminated individual.*

Description: Information system-related property includes, for example, hardware authentication tokens, system administration technical manuals, keys, identification cards, and building passes. Exit interviews ensure that individuals understand any security constraints imposed by being former employees and that proper accountability is achieved for all information system-related property. Exit interviews may not be possible for some employees (e.g., in the case of job abandonment, some illnesses, and non-availability of supervisors). Exit interviews are important for individuals with security clearances. Timely execution of this control is particularly essential for employees or contractors terminated for cause.

The following gives a definition of management control, a class of security controls.

Definition 5. *Management controls are defined as the safeguards and countermeasures for an information system that focus on management of risk and the management of information system security [29].*

The following example shows a management control from [29] which belongs to a family of System and Services Acquisition.

Example 2. *Control: Allocation of Resources*

Compliance Requirements:

- *Includes a determination of information security requirements for the information system in mission/business process planning.*
- *Determines, documents, and allocates the resources required to protect the information system as part of its capital planning and investment control process.*

- *Establishes a discrete line item for information security in organizational programming and budgeting documentation.*

Description: None.

The following gives a definition of technical control, a class of security controls.

Definition 6. *Technical controls are defined as the safeguards and countermeasures for an information system that are primarily implemented and executed by the information system through mechanisms contained in the hardware, software, or firmware components of the system"[29].*

The following example shows a technical control from [29] which belongs to a family of System and Communication Protection.

Example 3. *Control: Public Access protection.*

Compliance Requirements: The information system protects the integrity and availability of publicly available information and applications.

Description: The purpose of this control is to ensure that organizations explicitly address the protection needs for public information and applications with such protection likely being implemented as part of other security controls.

4.2.2 Control Extraction and Classification

Control Extraction and Classification is performed at design time. In this process, the controls that are applicable for the underlying system are extracted from the regulation. This also includes extracting the compliance requirements specified in controls. The extracted controls are then classified to determine the percentage of automation of regulation for the underlying system. The environment should monitor the target systems, target resources, target users, and target events in order to ensure automation of security controls. The target users are the users defined by the regulations. For example, privileged users, database administrators, security personnel etc., are the examples of target users. Target systems and the target resources are those to which the regulation applies. Examples of target systems are databases, applications, networks, firewalls etc. Target events are the events specified in security controls where an *event* can be defined as an occurrence at any point of time that can change the state of a system. The Compliance requirements specified in the controls provide us with the *compliant values* that a target user or a target resource must have to stay in compliance with the control.

System monitoring in z/OS is done using an external security manager (ex: RACF)

as discussed in Chapter 3. RACF facilitates logging security related events, such as users signing on and signing off, the issuing of RACF commands, and attempts to access protected resources. Generally, four main event types are required to be monitored.

- people related events: These events are related to the actions of employees or people such as security personnel, database administrator, system programmer etc., in an organization. The actions include but not limited to changing password, sharing user id's, access to unauthorized resources, successful and unsuccessful sign-on attempts, rejected access attempts to all resources, changes in the access control system that are carried out by an administrator, access attempts to the database that contains passwords and other sensitive information, activities performed by security personnel, system administrator, system programmer and others.

- asset related events: These events are related to the assets of an organizations such as access to the secure areas of an organization and access to the areas that contain information and information processing facilities.

- device related events: These events are related to the devices of an organization such as disposal of media, testing of back up copies of information, and protecting printed output etc.

- network related events: These events are related to network of an organization such as configuration of firewall, filtering between screened subnets and non-secure networks, packet filtering between secure networks and screened subnet, proxy servers, and network based intrusion detection etc.

All the security related events are recorded in different records and can be used in the program to perform meta-reporting on these reports. The next step after extracting the controls, is to classify them for determining the percentage of automation of regulations in the underlying system. An event based methodology is introduced to classify the security controls of any regulation. According to this methodology, the automation of a security control depends on the capability of a system to record and collect the information regarding the events specified in the control. The methodology classifies security controls into four classes as follows:

- Security controls that can be automated fully: A security control can be automated fully, if all the target actions/events specified in the control are observable/logged by the system.

- Security controls that can be automated partially for a system: A security control can be automated partially, if only a fraction of target actions/events are observable/logged by the system.

- Security controls that can be automated in future: A security control can be

automated in future, if there is a possibility to introduce a mechanism that can collect information of all the target events specified in the control.

- Security controls that cannot be automated: A security control cannot be automated if none of the target events/actions specified in the control are observable/logged by the system.

Example 4. *Control: The PASSWORD(INTERVAL) SETROPTS value is not set to 60 days.*

Description: INTERVAL specifies the maximum number of days that each user's password is valid. When a user logs on to the system, RACF compares the system password interval value specified in the user profile. RACF uses the lower of the two values to determine if the user's password has expired.

Compliance Requirements:

1. *If the PASSWORD(INTERVAL) value is set properly then the message "PASSWORD CHANGE INTERVAL IS xxx DAYS.", where xxx is less than or equal to 60 and not equal to 0, there is NO FINDING.*

2. *If the PASSWORD(INTERVAL) value is set improperly then the message "PASSWORD CHANGE INTERVAL IS xxx DAYS.", where xxx is greater than 60 or equal to 0, this is a FINDING.*

The above example is taken from the regulation, DISA STIG for z/OS [9] and is intended to check if the INTERVAL field of SETROPTS in RACF is set to 60 days. The control is extracted from the regulation DISA STIG and compliance requirements are extracted as shown in Table 4.2: If all the target events specified in the control are logged by the

Target user	none
Target resource	Password of a user
Target event	Setting password change interval
condition/Compliant value	Password change interval less than or equal to 60 and not equal to 0

Table 4.2: Control Extraction

system, the control can be automated fully.

4.2.3 Test Compliance

This process involves designing a generalized assessment routine for testing compliance of a system. Since, regulations are often formulated in different structures, general assessment routine must be designed in such a way that it can check compliance against any regulation. This generalized assessment routine can be used to develop a specific assessment routine for testing compliance against a particular regulation. Developing of the assessment routine is done using a compliance checking language known as Common Auditing and Reporting Language (CARLa) in this thesis. The assessment routine is then executed to test compliance of a system and produce compliance results. Compliance results include the *actual values* and the *compliant values* of an object in the domain. An actual value is a value of an object currently existing in the system whereas compliant values originate from regulation. The compliant values may be a numeric value, a classification type, process, and privileges etc. If the actual values does not match with the compliance values, it is said to be a *deviation*. Deviations can be found in this process by examining the compliance results that give a summary of compliance status of all the objects across multiple domains in the system.

Note: The design of assessment routine is done only once at design time whereas development of assessment routine is done at run time and is subjected to change as regulations are modified.

4.2.4 Reporting

A compliance report is a document comprising the results of compliance checking process. The compliance results convey the compliance status of a system with the considered regulations. The compliance results in the thesis provides the total number of compliant and non compliant rules in a regulation. In addition, the results also provide the number of compliant and non-compliant objects in a domain, their actual values, compliant values, rule name, domain name, and compliance status of each object. The total number of compliant objects in a domain with a rule or a control, is referred to as *compliance-number*. The compliance-number helps in tracking the compliance progress of the system. Presenting the actual values and compliant values of objects in the report makes deviation analysis faster.

Suppose, there are totally 100 users in the system out of which only 90 user's passwords are in accordance with the control specified in Example 4, then the compliance-number would be 90. If the organization succeeds to prove compliance for five more user's passwords, then the compliance number would be 95 and hence the compliance progress is 5.

4.2.5 Deviation Analysis

The compliance results are examined to check whether there are any violations after generating a compliance report. The system is said to be in compliance with the regulation if the actual values in the system match with the compliant values specified in the security controls. Deviation analysis aims to analyse the compliance results to find the deviations present in the system. This helps in early detection of deviations present in the system which is essential for improving the compliance status in the next iteration. If there are any deviations in the system, they will be handled by taking appropriate actions in deviation handling, otherwise the compliance management iteration ends.

Example 5. *Control: The PASSWORD(HISTORY) SETROPTS value is not set to 10.*

Description: HISTORY specifies the number of previous passwords that RACF saves for each USERID and compares with an intended new password. If there is a match with one of the previous passwords, or with the current password, RACF rejects the intended new password.

Compliance Requirements:

1. *If the PASSWORD(HISTORY) value is set properly then the message "x GENERATIONS OF PREVIOUS PASSWORDS BEING MAINTAINED.", where x is greater than or equal to 10, this is not a violation.*

2. *If the PASSWORD(HISTORY) value is set improperly then the message "x GENERATIONS OF PREVIOUS PASSWORDS BEING MAINTAINED.", where x is less than 10, this is a violation.*

The above example is taken from the regulation, DISA STIG for z/OS, [9]. In this, the compliant value is 10. Hence, the compliance report is analysed to check if the actual values of HISTORY field of all the objects in the domain is 10. If the actual value deviates from the compliant value, it is said to be a violation. Deviation Analysis aims to analyse all such deviations/violations existing in the system. This helps in determining the source of non-compliance thus making it easy for handling deviation.

4.2.6 Deviation Handling

The results of deviation analysis are fed as input to this process. Based on the results of deviation analysis, appropriate actions will be taken by the organization, either manually or in an automated way to fix the deviations in the system. In this process the actual values are set to match the compliant values. This process helps in achieving compliance

progress in the next iteration of compliance management.

If the results of deviation analysis consist of any deviations with respect to the control presented in Example 5, organization will try to address this by setting the actual values as specified in the compliance requirements either manually or in an automated way. This improves the compliance status of the system in the next iteration of compliance management.

Chapter 5

Implementation of Automated Compliance checking for z/OS

This chapter discusses about the implementation of an automated compliance checking solution for z/OS using the proposed methodology presented in Section 4.1. *Compliance checking* is defined as the problem of verifying whether or not a given system fulfils the compliance requirements of a regulation. The regulations against which compliance is being checked are Global Services Division (GSD 331) and Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG). This chapter also gives a brief description of GSD 331 and DISA STIG and presents the classification of controls of each regulation. The methodology proposed for implementing an automated compliance checking process involves the following steps:

- Regulation Analysis.
- Control Extraction and Classification.
- Test Compliance.
- Deviation Analysis.
- Deviation Handling.
- Reporting.

However, Deviation Analysis, Deviation Handling, and Reporting are out of the scope of automated compliance checking process. Hence, we focus only on Regulation Analysis, Control Extraction and Classification, and Test Compliance in this thesis.
The remainder of the chapter is structured as follows: Section 5.1 gives a brief description

of GSD 331 and DISA STIG along with the analysis of these regulations for automation of security controls in z/OS. Section 5.2 discusses about the control extraction and classification of DISA STIG and GSD 331. Section 5.3 presents the architecture of automated compliance checking solution for z/OS. In addition, it presents the design of a general Assessment Routine for automated compliance checking. This is implemented using CARLa (compliance checking language used in the thesis). An introduction to CARLa is given in Section 5.3.1. Section 5.3.3 presents the compliance checking of z/OS against a sample regulation which consists of six rules. Finally, Section 5.3.3 presents the compliance results.

5.1 Regulation Analysis

The objective of regulation analysis is to analyse the regulation, understand the structure of controls, and determine various factors such as domains of the controls, control families, compliance requirements of each control, roles and resources defined in the regulation.

5.1.1 GSD 331

GSD 331 [8] is the IBM's primary information security controls documentation for strategic outsourcing customers. GSD stands for Global Services Division. The more recent version of GSD 331 is Information Security Controls (ISeC). GSD documents the company security requirements, so that IBM can implement them on the company's systems and services. These requirements include defined roles, responsibilities and specific controls. The GSD 331 only covers IT security controls, not application development, system management, or Disaster Recovery (DR). The version of GSD 331 used in the thesis is 6.0a and it consists of 64 security controls in total.

5.1.2 DISA STIG

Security Technical Implementation Guide (STIG) [9] is a methodology for standardized secure installation and maintenance of computer software and hardware. The term STIG was coined by DISA (Defense Information Systems Agency). DISA is the United States Department of Defense agency that provides information technology (IT) and communications support to the President, Vice President, Secretary of Defense, the military Services, and Combatant Commands. DISA also creates configuration documents in support of the United States Department of Defense (DoD). The guidelines include recommended administrative processes and span over the lifecycle of the device. STIGs are

used to maintain availability, confidentiality and integrity of information systems and also configuration management of the system.

The use of STIGs is beneficial in many situations. One of the examples where STIG is very beneficial is in the configuration of a desktop computer. Most operating systems do not posses strict authorization for the access of resources. This makes easy for malicious people to have an access to the system and thus affecting various security aspects such as availability, confidentiality, and integrity. The STIGs contain technical guidance to "lock down" information systems/software that might otherwise be vulnerable to a malicious computer attack. Also, STIG describes various rules for minimizing network-based attacks and also enforces strict authorization rules for computer criminals. STIG may also be used to describe the processes and lifecycles for maintenance such as software updates and vulnerability patches.

The most advanced use of STIG is in the design of corporate network. A corporate network may consist of thousands of network devices and servers that control the flow of information. In order to make the network efficient and secure, STIGs may also be used to define a common configuration for each device type such as routers, firewall, domain name servers, network protocols, and switches. The title, version, and release of the DISA STIG used in the thesis are z/OS RACF STIG, 6 and 8 respectively and it consists of 274 security controls or rules in total.

Following are the security control families that both DISA STIG and GSD 331 have in common. However, the compliance requirements are not same in the controls of each family.

1. **Physical access controls**
   Physical access to information processing resources expose them to several risks such as theft or damage of information processing resources, unauthorized disclosure or erasure of information, and interruption of support for business processes. Hence, management should establish these physical access controls to address the above problem. This section will address controls to protect data centers under IBM's control, access to the IBM controlled data centers, printer security, storage media, and LAN infrastructure control.

2. **Logical access controls**
   This category of controls will address several aspects of logical access control. The access being controlled is to either host or LAN systems, data or system privileges. Logical access controls include the following primary topics.

   - **Identify and Authenticate Users:** Ensure that a unique identifier (e.g., userid) which shows valid overall ownership to an individual associated with each potential user of the system. When the user enters the system, ensure that a further level of identification (e.g., a password) verifies that the user is

who he/she claims to be.

- **Define and Protect Resources:** Ensure that each resource on the system can be identified, that access to the resource can be allowed at the appropriate levels for authorized users, and that access can be denied for unauthorized users.

- **System and Security Administration:** Ensure that only authorized users can set, modify, or disable system security functions.

- **Log Access Attempts:** Ensure that an audit record can be created for each successful or unsuccessful access attempt to the system or to protected resources on the system.

- **Report Access Violations:** Ensure that unauthorized access attempts to systems or information can be recognized as violations, either immediately or on subsequent analysis.

3. **Portable storage media**
   To prevent unauthorized disclosure, modification, removal or destruction of assets, and interruption to business activities, Media should be controlled and physically protected. Appropriate operating procedures should be established to protect documents, storage media (magnetic tape and removable and non-removable optical or magnetic disks and cartridges, and system documentation from unauthorized disclosure, modification, removal and destruction. This controls includes Media labeling (labels can be placed outside of portable media designating ownership or data classification), Physical protection of storage media, Custodial media Inventory control where custodial media is portable media that is not backup media or vendor media(sent by software/hardware vendor to distribute their products).

4. **Security/Integrity Advisory Process**
   A Security/Integrity Advisory is a warning of an exposure in a program or process that allows unauthorized users to gain privileged authority on a system, to bypass access controls, or to gain unauthorized access to data. A Security/Integrity Advisory process should be followed to install the fixes. The core requirements for this process are:

   - Determination of risk severity based on vulnerability rating and exploitation category.

   - Notification of fix availability

   - Procedure to determine the schedule for application of the security/integrity fixes. Only advisories with available fixes will be installed.

- The process is auditable.

5. **Network controls**
   Networks should be adequately managed and controlled, in order to be protected from threats, and to maintain security for the systems and applications using the network, including information in transit.

6. **Firewalls**
   Firewalls are systems designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both. Firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

GSD 331 address the following security control families in addition to the above families.

1. **Security status checking**
   The health checking process must be conducted on a periodic basis to verify the security status of the systems. A process must be in place to perform security health checks of managed systems and systems software to validate compliance with the Technical Specifications listed in Customer XYZ implementation manual. Management must decide on the level of Security Health Checking based on the control necessary for the customer account or device. The following are the examples of security controls existing in GSD 331 which are related to user password management. These security controls ensures that the passwords in the system are robust and the format of the password is according to the regulation.

2. **Application/End User security**
   The security of the programs and data associated with applications should be protected from unauthorized access and use. In order to accomplish the protection, the following specifications should be obtained from the application owner:

   - Public access for the programs and data
   - List of users who need additional access to the programs and data
   - Auditing requirements for the programs and data

3. **Security incident management**
   A security incident can originate within or outside our work environment, can involve external sites, and can range in severity. IT security incidents potentially involve system penetrations, destruction of data, fraud, crime or other serious matter. Other incidents can generally be referred to site management and personnel

for resolution. This section deals with management of such security incidents.

One of the common security control in GSD 331 and DISA STIG is presented in the following example which belongs to security control family of Logical access controls.

Example 6. *Control: Maximum change interval of the password should be limited.*

Description: Change interval of password specifies the maximum number of days that each user's password is valid.

Compliance Requirement of STIG: If the password change interval is set to less than or equal to 60 days and not equal to zero, it is not a violation.

Compliance Requirements of GSD 331: If the password change interval is set to less than or equal to 90 days, it is not a violation.

From Example 6, we can notice that though the controls are common, regulations differ in their compliance requirements.

5.2 Control Extraction and Classification

To facilitate the automation of security controls, there are 'fields' defined in zSecure, which can be used in assessment routine for checking a value of a particular resource in z/OS. For ex: there exists a field known as HISTORY to check the value of password history in the system. Security controls are classified into different categories based on the existence of the field for a particular control. The classification methodology shown below is an instantiation of the general classification methodology presented in Section 4.2.2.

1. Security controls that can be fully automated: A security control can be automated fully if there exists fields for all the target events specified in the compliance requirements of a security control.

2. Security controls that can be partially automated: A security control can be automated partially if there exists fields in zSecure for some of the target events specified in the compliance requirements of a security control.

3. Security controls that cannot be automated: A security control cannot be automated if there are no fields defined in zSecure for any of the target events specified in the compliance requirements of a security control.

4. Security controls that can be automated in future: A security control can be automated in future if there is a possibility of adding a new field into zSecure to check the value of target events specified in the compliance requirements of a security control.

Table 5.1 presents a part of the classification of GSD 331 and DISA STIG which was prepared as part of the classification task. In order to classify a control into one of above classification types, the controls and their compliance requirements must be understood clearly and zSecure field names required to check the control must be determined. Table 5.1 has four different columns namely, *Base control* is defined as the control statement with no specific compliance requirements, 'GSD 331 ID' is the rule or control ID of GSD 331 control, which is similar to the Base control, 'DISA STIG ID' is the rule or control ID of DISA STIG control, which is similar to the Base control, and 'z/OS Automation' classifies the control into one of the above classification types of automation.

A base control is defined by studying all the regulations and extracting the common security controls existing in both the regulations. GSD 331 ID and DISA STIG ID are the Identification numbers of controls which are selected by comparing each control of the regulations to the base control and extracting the one which is similar to the base control. Finally, z/OS automaton is determined by examining if the fields required to check the controls are defined in zSecure or not and is based on the methodology discussed in the above paragraphs.

Sl no.	**Base Control**	**DISA STIG ID**	**GSD 331 ID**	**z/OS Automation**
1	Security Perimeters (barriers such as walls, card controlled entry gates, or manned reception desks) should be used to protect areas that contain information and information processing facilities.	ZFEP0011: All hardware components of FEPs are not placed in secure locations where they cannot be stolen, damaged, or disturbed.	1.1.2.1: is same as the base control.	The controls cannot be automated.
2	Maximum change interval of password should be properly defined.	RACF0440: The password interval setropts value is not set to 60 days.	2.1.2: Maximum change interval of password should be 90 days.	The controls can be automated fully.

3	User ID revalidation of continued business need.	None	A documented process should be in place for revalidation of userids. This revalidation should review the userids and determine if there is still a business need for the user to access the system. Users that no longer require access should be reported to the userid administrator for deletion.	Control can be automated in future.
4	RACF batch jobs are not protected with propagation control.	RACF0600: 1. The PROPCNTL resource class is active. 2. A PROPCNTL resource class profile is defined for each userid associated with a job scheduler (eg., CONTROL-M, CA-7, etc.) and a Multiple User Single Address Space Systems (MUASSS) able to submit batch jobs. If both of above conditions are true, this is not a violation.	None	Control can be automated partially.

Table 5.1: Classification of GSD 331 and DISA STIG

The controls ZFEP0011 and 1.1.2.1 of DISA STIG and GSD 331 cannot be automated as there is no database which stores information related to physical location and attributes of that location in z/OS. The automation of controls RACF0440 and 2.1.2 depend on the existence of a field to check the password change interval value. Since, there exists a field known as INTERVAL in zSecure that stores information regarding password interval, the controls can be automated fully.

In order to automate the third control, there should be an interface which can collect the information regarding existence of documented process for revalidation of userids. Since, there is a possibility of collecting this information by adding a Tivoli Identity Manager (TIM) interface which provides user id recertification functionality, the control can be automated in future.

In order to automate the security control RACF0600 completely, it should be possible to check both the conditions specified in STIG control. Conditions that must be checked here are is if PROPCNTL resource class is active and if PROPCNTL class is defined for each userid. Since, only the first condition can be checked with the help of a field known as PROPCNTL, and since information about userids associated with CA-7 are not collected, only the first condition can be tested. Hence the control can be automated partially.

There are totally 339 security controls from both GSD 331 and DISA STIG out of which 140 security controls can be automated fully, 69 controls cannot be automated for z/OS, 96 controls can be automated in future, and 33 controls can be automated partially for z/OS. Figure 5.1 gives a graphical representation of the results obtained from classification of STIG and GSD 331 controls respectively.

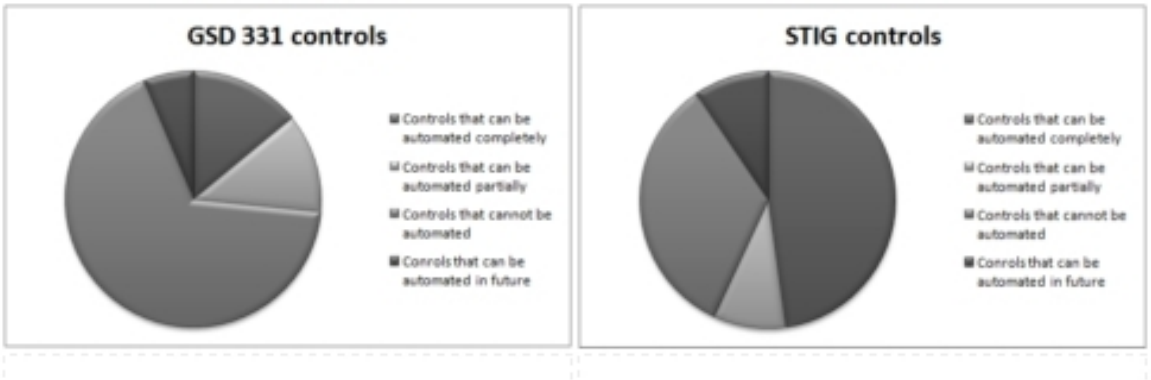


Figure 5.1: Automation of DISA STIG and GSD 331 for z/OS

5.3 Test Compliance

Figure 5.2 depicts the architecture of an automated compliance checking for z/OS. The core of the automated compliance checking process is the Assessment Routine. An assessment routine is a fixed code or program developed to assess a system for compliance. One of the main goals of the thesis would be to develop this assessment routine for implementing an automated compliance checking process for z/OS. CKRCARLA is the

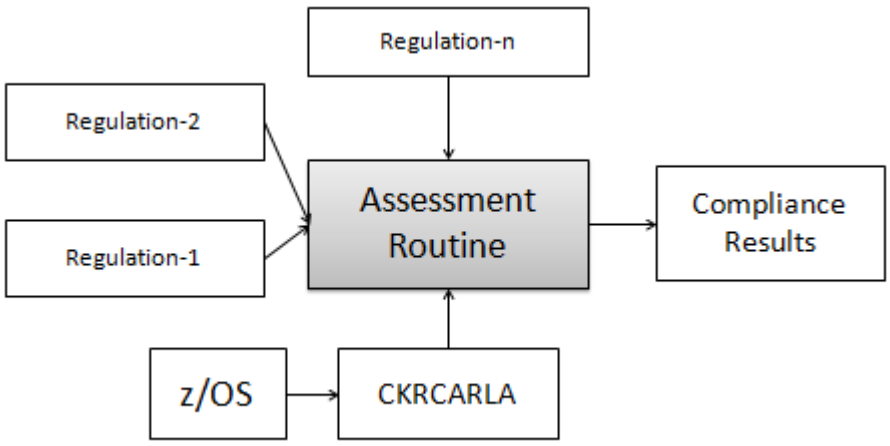


Figure 5.2: Automated Compliance checking of z/OS

main program of zSecure Admin and Audit which performs the following:

- Interprets the CARLa code.
- Collects the data from the security database, CKRFREEZE files, SMF, and other data.
- Generates reports based on the data collected.

The various data collected different databases such as security database, CKFREEZE files which consists of all current settings in the system, SMF data which consists of all logged security data, and other data from the CKRCARLA along with the compliance requirements of the regulations are fed as input to the assessment routine. Assessment routine compares the compliant values in the regulations with that of the actual values present in the system to check for compliance. If the actual value match the compliant value specified in the regulation, the corresponding resource is in compliance with the security rule or the control. The result of the assessment routine is the summary of compliance status of all the resources in the system against the corresponding security controls of the regulations.

The programming language, CARLa has been used to implement the assessment routine. CARLa is the auditing and the reporting language used in z/OS. A brief introduction to CARLa is given in Section 5.3.1.

5.3.1 Introduction to CARLA

CARLA [7] is the main reporting engine that is used within IBM Security zSecure. CARLa stands for Common Auditing and Reporting Language. The language is sophisticated and is being used to generate reports, emails, XML, as well as actions. Examples of possible actions include RACF, TSO, UNIX, and IDCAMS commands, or CARLa code. CARLa can process huge amount of data in an efficient way using only a small amount of CPU cycles. The language contains several features that normally exist in a data processing language including statistical and mathematical features. CARLa is repeatable, scalable and structured. For example,

- A group of CARLa commands can be built to help with or automate administration of security administration and auditing tasks.
- CARLa programs can be re-used on any system where IBM Security zSecure is installed with minimal or no changes to the corresponding CARLa code.
- CARLa programs written for SMF event reporting in zSecure Audit can also be used by zSecure Alert for real-time SMF event reporting.
- CARLa programs can be written such that it produces output in a variety of formats including ISPF, print, WTO, email, or XML.

CARLA can take various data sources as input to generate reports such as RACF data, SMF data, CKFREEZE data, access monitor data, and external files. For example, data from RACF database (active, unloaded, or copied data), SMF data (active and dumped data), and CKFREEZE file (system configuration) can be correlated using one CARLa program. Examples of external files are data extracts from a Human Resources system, unloaded data from DB2 tables, unloaded tables or databases containing a list of user IDs, and unloaded log files. Figure 5.3 depicts the input sources supported by CARLa. Table 5.2 gives a brief description of these data sources. Input files are allocated using SETUP FILES command or option SE.1 in the zSecure Admin or zSecure Audit ISPF session or using an ALLOC statement. An ALLOC statment is used to select a set of security databases and CKFREEZE, SMF, or command output files. Table 5.3 gives an overview of CARLa commands. Example 7 shows a CARLa code to extract all user IDs with the mask of ITSO.

Example 7. `newlist type=RACF`

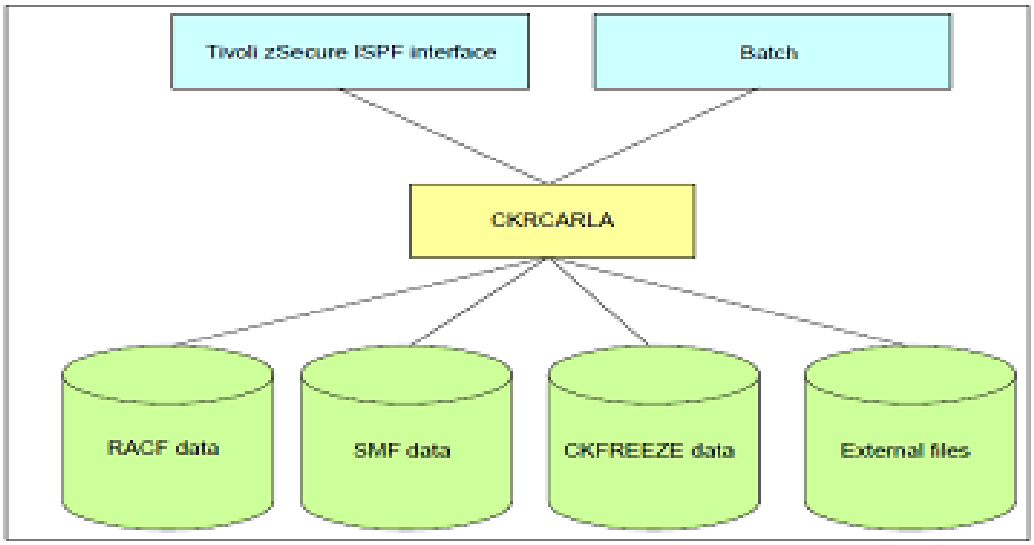


Figure 5.3: Data sources supported by the CARLa program

Data source	**Description**
RACF data	Supported RACF data includes the active primary or backup RACF database, An unloaded RACF database unloaded by zSecure Admin (CKRCARLA), a RACF database copy taken with the standard IBM RACF utilities IRRUT200 or IRRUT400.
SMF data	Supported SMF data includes the active SMF datasets, SMF records in their original format dumped by the IBM utility IFASMSDP or the zSecure Audit utility C2RJFUNL.
CKFREEZE data	Supported CKFREEZE data includes Live settings based on information directly read from the control blocks of the current live system, A dataset containing resource information gathered by the IBM Security zSecure Collect job.
Access monitor data	Access datasets are generated by the Access monitor Started task. They are a collection of all RACF acces decisions that can be used both for access reporting and cleaning up access control lists or entire resource profiles.
External files	These are installation defined datasets, which can be referred in CARLa programs. Prerequisite for using external files in CARLa is that all the variables must be defined.

Table 5.2: Datasources supported by CARLa

```
select class=USER segment=BASE mask=ITSO*
sortlist key(8) name dfltgrp owner
```

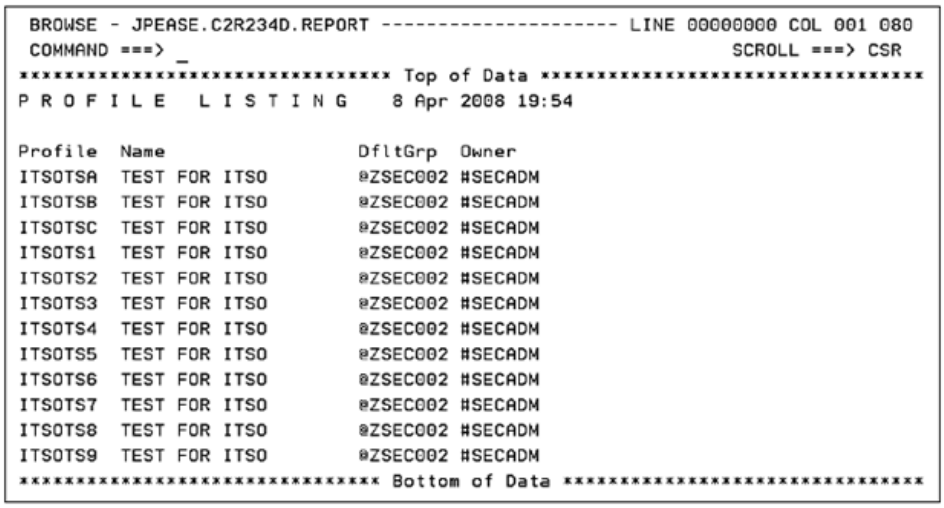
CARLa Statement	**Description**
NEWLIST	Marks the start of a new CARLa report.
TYPE	Select the type of NEWLIST to be used as an input for the NEWLIST. For example, TYPE=RACF is used to extract and report RACF information. The RACF NEWLIST is the default if TYPE statement is omitted. There are around many other NEWLIST types (around 58 for RACF systems) that can be used to generate specific reports.
SELECT	This statement is used to select/filter records that match all the conditions specified in the SELECT statement.
CLASS	Select the RACF profile class to report on.
SEGMENT	Select the segment to report on from the RACF profile (For example, BASE, TSO, CICS, or OMVS).
MASK	Select profiles using a mask (wildcard) on the profile key.
SORTLIST	Specify the fields to be shown for the selected records in the report based on the selection criteria.
KEY	The profile names found that match the selection criteria.
DFLTGRP	The default group of the pertinent user profile.
OWNER	The owner of the profile.
/*...*/	Comments in the CARLa code.

Table 5.3: An overview of CARLa statements

The above example extracts all user IDs with the mask of ITSO and produces a report listing all user IDs matching this mask. The newlist type selected in the example is RACF, thus the code reports on RACF information. The RACF profile selected is the USER and segment being selected is BASE. The sorlist in the example are key, name, default group, and owner. Hence these fields will be displayed in the report. The report produced upon executing the above CARLa program is shown in the Figure 5.4.

5.3.2 Assessment Routine

Assessment routine is the core of automated compliance checking. An assessment routine is a fixed code or program which can assess the compliance of a system. In order to implement the Assessment Routine, we have extended CARLa by introducing a new CARLa statement called STANDARD through which we can define a standard or the regulation and a newlist type, TYPE=COMPLIANCE to perform “meta-reporting” of the results of testing data from other newlist types. The STANDARD consists of a set of security control or RULE definitions which have a rule name and description that both originate from the standard/regulation, and a DOMAIN of objects to which the



```
 BROWSE - JPEASE.C2R234D.REPORT ---------------------- LINE 00000000 COL 001 080
 COMMAND ===> _                                          SCROLL ===> CSR
******************************** Top of Data ********************************
P R O F I L E   L I S T I N G    8 Apr 2008 19:54

Profile  Name                 DfltGrp  Owner
ITSOTSA  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTSB  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTSC  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS1  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS2  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS3  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS4  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS5  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS6  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS7  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS8  TEST FOR ITSO        @ZSEC002 #SECADM
ITSOTS9  TEST FOR ITSO        @ZSEC002 #SECADM
******************************* Bottom of Data ******************************
```

Figure 5.4: Report produced by the example CARLa program

rule applies, and a set of TEST specifications that must all be satisfied for the object to be in compliance with the rule. A TEST field in the Assessment routine helps to specify a test condition of each rule or control. A TEST specification consists of a test variable which is a zSecure field name, a relational operator for specifying the compliance condition, and a compliant value that originates from the regulation. The Assessment Routine used for automated compliance checking of z/OS is given below.

```
STANDARD <standardname> DESC(<description>) VERSION(<version>)

DOMAIN<domainname> SELECT(<type>[(<selclause>)]),
RULE <rulename> DESC(<desc>) DOMAIN(<name>)
TEST <testname> <type>(<fieldname><reloper><compliantvalue>)
ENDRULE

DOMAIN<domainname> SELECT(<type>[(<selclause>)]),
RULE <rulename> DESC(<desc>) DOMAIN(<name>)
TEST <testname> <type>(<fieldname><reloper><compliantvalue>)
ENDRULE
.
.
DOMAIN<domainname> SELECT(<type>[(<selclause>)]),
RULE <rulename> DESC(<desc>) DOMAIN(<name>)
TEST <testname> <type>(<fieldname><reloper><compliantvalue>)
ENDRULE

ENDSTANDARD
```

```
SUPRESS STANDARD=<name> RULE=<name> REASON='<Description>'
newlist type=compliance
sortlist field 1 field2 .. filed n
DEFINE cnt_cmply sumcount
DEFINE cnt_noncomply sumcount
SUMMARY field1, field2,..fieldn
```

STANDARD statement in the above routine helps defining a standard/regulation. In addition, version and the description of the regulation can be written with the help of VERSION and DESC statements respectively. Any regulation consists of set of rules or security controls which further consist of several TEST conditions in order to check the controls. RULE statement in the routine helps to define a rule along with its name, description. The TEST statement of the routine allows specifying the test condition or compliance requirements with the help of field name, relational operator, and compliant value.

The rules/controls may apply on different domains, DOMAIN statement in the routine supports defining a domain to which the rule applies. Every rule has an associated domain. However a domain can apply on a set of rules. In this case, domain is defined only once and this domain apply to all the rules. NEWLIST statement generates a compliance report with the fields specified in the sortlist. DEFINE statements are used to define the sum count of compliant objects and non-compliant objects. SUMMARY statement is similar to sortlist statement. SUPPRESS command can be used to suppress counting the compliance or non-compliance of the indicated rule to the compliance results. The REASON keyword supports describing the reason for suppressing a particular rule. It allows to specify the fields to be displayed in the results. Description of all the statements and fields used in the routine is presented in Appendix A. For an object to be in compliance with the control/rule, it should satisfy all the TEST conditions specified in the rule.

5.3.3 Verification

The above assessment routine is used in automated compliance checking of z/OS against a sample regulation. A standard called SAMPLE is defined which consists of 6 Rules of DISA STIG and GSD 331. z/OS is checked for compliance against all these rules and compliance results will be generated which consists of compliance status of z/OS against the regulation.

1. The first rule checks if the /tmp directory is having a sticky bit assigned.

2. The second rule checks if the tmp is a symlink to the $SYSNAME/tmp directory.

3. The third rule checks if the users associated with the started tasks have the PRO-

TECTED attribute and OWNED by STCGROUP.

4. The fourth rule checks if uacc option for sensitive datasets is less than update, success audit less than or equal to update, and.failure_audit has read option set.

5. The fifth rule checks if update-sensitive data sets has only UPDATE for SYSPROG.

6. The sixth rule checks if the universe to test, e.g. all users that have SPECIAL are owned by group SYSADM and have connect group SECADM or RACFADM, the exceptions, e.g. IBMUSER is also allowed to have SPECIAL privilege.

```
STANDARD SAMPLE

/*Rule 1: "the /tmp directory must have the sticky bit assigned" */

DOMAIN unix_tmp SELECT(UNIX(fs_mountpoint='/' depth=0 filename='tmp' ))
RULE unix_tmp_sticky DOMAIN(unix_tmp)
TEST sticky(UNIX(attr="o+t"M))
ENDRULE

/* Rule 2: "tmp is a symlink to the $SYSNAME/tmp directory" */

RULE unix_tmp_symlink DOMAIN(unix_tmp)
TEST symlink(UNIX(SYMLINK))
TEST $sysname(UNIX(LINK_TARGET=' $SYSNAME/tmp'))
ENDRULE

/* Rule 3: "users associated with started tasks must have the PROTECTED
attribute and OWNED by STCGROUP" */

DOMAIN stc SELECT(R_STC)
RULE stc_user_def DOMAIN(stc)
TEST protected R_STC(userid:protect=yes)
TEST owner_stcgroup R_STC(userid:owner=STCGROUP)
ENDRULE

/* Rule 4: checks if uacc option for sensitive datasets is less than update, success
DOMAIN APFlibs SELECT(SENSDSN(APF=yes) R_SENSITIVE)
RULE APF_profile DOMAIN(APFlibs)
TEST uacc R_SENSITIVE(UACC<UPDATE)
TEST success_audit R_SENSITIVE(SAUDIT<=UPDATE)
TEST failure_audit R_SENSITIVE(FAUDIT=READ)
ENDRULE
```

```
/* Rule 5: "update-sensitive data sets only UPDATE for SYSPROG" */

DOMAIN RACF_update_sensitive_permits
SELECT(SENSDSN(RISK=UPDATE not (sensitivity=:site)) RACF_ACCESS(access>=UPDATE))
RULE update_sensitive_only_update_SYSPROG DOMAIN(RACF_update_sensitive_permits)
TEST update_sysprog RACF_ACCESS(id=sysprog)
ENDRULE

/* Rule 6: the universe to test, e.g. all users that have SPECIAL
- these must all be owned by group SYSADM and have connect group SECADM or RACFADM
- the exceptions, e.g. we also allow IBMUSER to have special  */

DOMAIN RACF_system_special SELECT(racf(c=user s=base spec))
DEFINE type=racf special_required_connect boolen where cggrpnm=(sysadm,racfadm)
RULE RACF_special_def DOMAIN(RACF_system_special) EXEMPT(racf(key=IBMUSER))
TEST required_owner RACF(owner=sysadm)
TEST required_connect RACF(special_required_connect=yes)
ENDRULE

newlist type=compliance, name=check
sortlist ' object' class(0) resource(0) /,
'test' test_name(0),
test_newlist_type(0) | '(' |,
test_field(0) | test_reloper(0) | test_compliant_value(0) | ')',
'value=' | test_field_value(0) 'result=' | test_result(0),
'(base' test_field_base_value(0) | ')',
'exempt=' | rule_exempt(0)
/* rule_exempt(hb,'exempt',0) */
define cnt_comply sumcount where test_compliant=yes
define cnt_noncomply sumcount where test_noncompliant=yes
summary / 'Standard' standard(0) standard_desc(0) *,
/ 'rule' rule(0),
'compliant=' | cnt_comply(0),
'non-compliant=' | cnt_noncomply(0),
rule_desc(0),
/ 'domain' domain_name domain_desc(0) /
```

Compliance Results

The above results are obtained by executing the implemented routine in one of the test z/OS systems, namely EZOS. EZOS has many applications, and subsystems installed in it such as Database Manager (DB2), Customer Information Control Systems

(CICS), Job Entry Subsystem (JES), Information Management Systems (IMS), Storage Management Subsystem (SMS), VTAM, Terminals, and Advanced Program to Program Communication (APPC) sessions etc. The events of all these applications are logged in various datasets as discussed in Chapter 3. The datasets that are selected as input for the assessment routine are active backup RACF database, live SMF datasets, and CK-FREEZE datasets. Hence, the assessment routine can check compliance on racf events, events logged to SMF and current system settings.

Compliance results include the following information: Regulation name, total number of compliant and non compliant rules/controls, Rule name, Compliance number, i.e., total number of objects in the domain that are compliant with the rule, total number of objects that are not compliant, Control/rule description, exemption objects, compliant value, test condition, actual value, test result which is the compliance status of the object with the rule, domain name, number of records processed, types of records processed, and total time used to check the compliance of the system.

Figures 5.5, 5.6, 5.7, 5.8, and 5.9 shows the results obtained from compliance checking of z/OS against the above regulation. Figure 5.5 shows the number of compliant and non compliant objects in the domain on which rule 3 applies. From Figure 5.5, we can notice that there are 202 compliant and 1548 non compliant objects in the domain. This means compliance number for rule 3 is 202. Figure 5.5 shows the total number of rules in the regulation that are in place. According to this results, out of six controls three controls are in place. Figure 5.6 shows the subset of objects and their compliance status with rule 3.

Figure 5.7 shows the compliance status of z/OS with rule 1 and rule 2. From this figure, we can notice that there is only one object present in the domain of rule 1 and is compliant thus compliance number equal to 1. In addition, there are two objects which are present in the domain of rule 2 and are non complaint. Figure 5.8 shows the compliance status of z/OS with rule 6. Figure 5.9 shows the number of records of different domains processed in order to generate the compliance results. From this figure, we can also see that the time taken to check the compliance of z/OS against this regulation is 0.35 seconds.

From the figures, we can notice that compliance results provide the total number of rules or controls that are in place, compliance number i.e., the total number of objects in a domain that are compliant and non-compliant, actual value denoted by 'value' field in Figures 5.6, 5.7, and 5.8, compliant value, rule name, description of the rule, exempt fields, and test results of each rule. Compliance number is very useful in tracking the compliance progress. Displaying actual value and compliant values of an object helps in deviation analysis and deviation handling thus making compliance management faster.

To the regulation specified above, fourteen more rules were added and checked for the compliance. Figures 5.10 and 5.11 shows the compliance status and time taken to check

Figure 5.5: Automated Compliance checking of z/OS-1

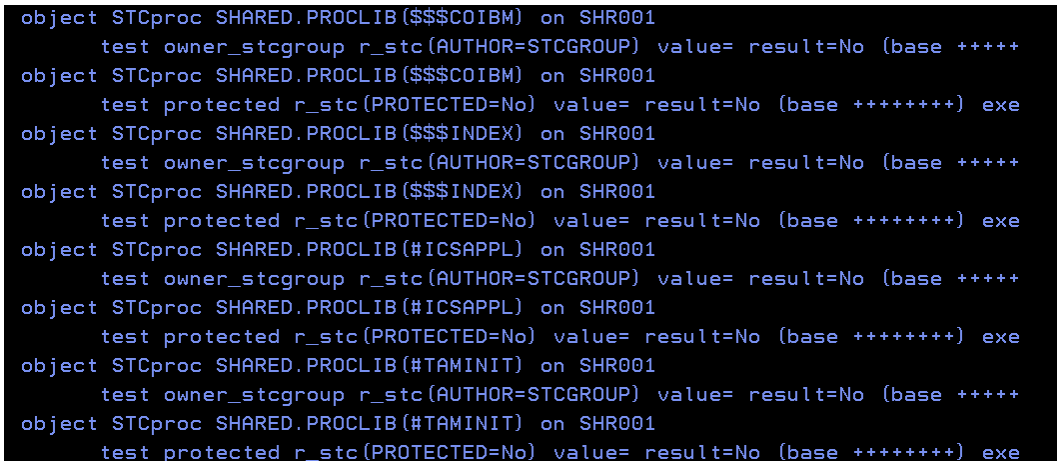


Figure 5.6: Automated Compliance checking of z/OS-2

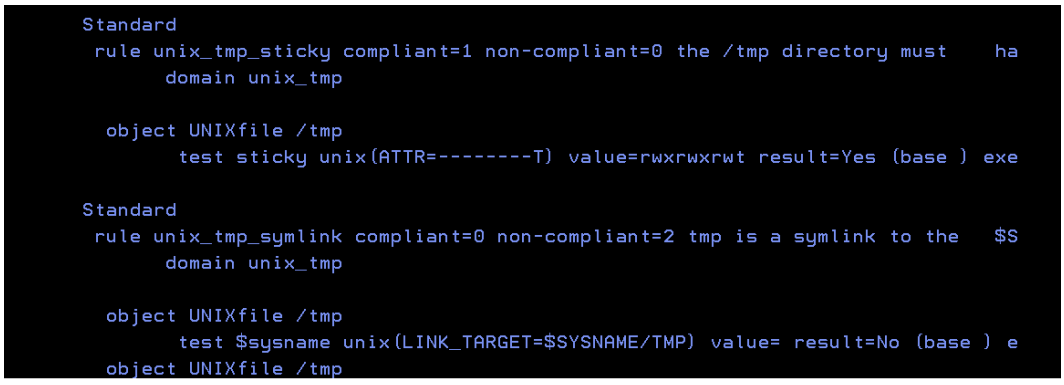


Figure 5.7: Automated Compliance checking of z/OS-3

the compliance of z/OS against modified regulation. From the Figure 5.11 we can notice that the time taken to check the compliance against twenty rules still remains same i.e., 0.35 seconds.

```
Standard
 rule NameIsEend compliant=1 non-compliant=0
       domain System

  object
       test NameIsEEND system(SYSTEM=EEND) value=EEND result=Yes (base ) exemp

Standard
 rule RACF_special_def compliant=0 non-compliant=14 ownership and connect requi
       domain RACF_system_special

  object USER CRMASC2
       test required_owner racf(OWNER=SYSADM) value=CRMA result=No (base ) exe
  object USER CRMBGP2
       test required_owner racf(OWNER=SYSADM) value=CRMB result=No (base ) exe
```

Figure 5.8: Automated Compliance checking of z/OS-4

```
  CKRCARLA TESTCMP1      38 sensdsn                        0 SYSPRINT
  CKRCARLA TESTCMP1      39 racf_access                10153 SYSPRINT
  CKRCARLA TESTCMP1      46 racf                          14 SYSPRINT
  CKRCARLA TESTCMP1      54 compliance                  1774 SYSPRINT      38

  CKR1137 00 1,768 compliance records processed, selected 1,768 (100%)
  CKR1137 00 359 r_sensitive records processed, selected 359 (100%)
  CKR1137 00 875 r_stc records processed, selected 875 (100%)
  CKR1137 00 9,472 racf records processed, selected 14 (0%)
  CKR1137 00 39,087 racf_access records processed, selected 10,153 (25%)
  CKR1137 00 1,598 sensdsn records processed, selected 0 (0%)
  CKR1137 00 1 system records processed, selected 1 (100%)
  CKR1137 00 104,478 unix records processed, selected 1 (0%)
  CKR0039 00 CKRCARLA used 0.35 CPU seconds, 4+17,148KB, and took 1.0 wall clock
             Region requested 0KB, granted 11,240+1,931,264KB
             max used in jobstep 652+23,828KB
```

Figure 5.9: Automated Compliance checking of z/OS-5

```
Standard SAMPLE
 rule  compliant=12 non-compliant=8
       domain


Standard
 rule inactivesetrop compliant=1 non-compliant=0 The inactive setropts value is
       domain SETROPTIONS
```

Figure 5.10: Addition of new rules to the regulation

```
SDSF OUTPUT DISPLAY CRMBGB1X JOB06513  DSID   102 LINE 4,841   COLUMNS 01- 80
COMMAND INPUT ===> _                                          SCROLL ===> CSR
CKRCARLA TESTCMP1      91 setropts                          1 SYSPRINT
CKRCARLA TESTCMP1     135 racf                             14 SYSPRINT
CKRCARLA TESTCMP1     143 compliance                     1823 SYSPRINT      18

CKR1137 00 1,803 compliance records processed, selected 1,803 (100%)
CKR1137 00 359 r_sensitive records processed, selected 359 (100%)
CKR1137 00 875 r_stc records processed, selected 875 (100%)
CKR1137 00 9,474 racf records processed, selected 34 (0%)
CKR1137 00 39,091 racf_access records processed, selected 10,154 (25%)
CKR1137 00 1,657 sensdsn records processed, selected 0 (0%)
CKR1137 00 1 setropts records processed, selected 1 (100%)
CKR1137 00 1 system records processed, selected 1 (100%)
CKR1137 00 104,521 unix records processed, selected 1 (0%)
CKR0039 00 CKRCARLA used 0.35 CPU seconds, 4+17,404KB, and took 1.0 wall clock
           Region requested 0KB, granted 11,240+1,931,264KB
           max used in jobstep 652+24,084KB
```

Figure 5.11: Time taken to check compliance of a z/OS system against modified regulation

5.4 Discussion

In Chapters 1 and 2, the limitations that current approaches are facing in automated compliance checking were identified. In this section, we discuss how the assessment routine presented in this chapter can address these shortcomings.

Multiple regulation types: Often, organizations are compelled to comply with multiple types of regulations. Current approaches [16] for compliance checking does not support different types of regulations in compliance checking as discussed in Chapter 2. This problem is addressed in the thesis by the use of CARLa member. The definition of a member in a library is discussed in Chapter 3. A member is a unit a record within a partitioned dataset or a library and a *CARLa member* is a record written in CARLa within a CARLa library in zSecure. A CARLa member can hold the carla code of a specific regulation for compliance checking. So, in order to have multiple regulations in compliance checking, multiple CARLa members will be created for each of the regulation. Compliance checking against multiple regulations can be done using IMBED statement in Job Control Language (JCL). IMBED statement in JCL allows specifying the input for the program execution. This allows one to check the compliance of a system against multiple regulations.

Multiple domains: Many methodologies [28], [17], [16] proposed for automated compliance checking fail to check the compliance with the rules across various domains. They are often domain specific. However, the routine used for automated compliance checking in the thesis addresses this problem using a CARLa statement known as DOMAIN, which allows one to define various domains in compliance checking. This allows one to

check compliance of a system across mulitple domains.

Flexibility: One of the efficient ways to implement an automated compliance checking tool is to support modularity in the design. The modularity in the code helps adapting to the changes in the regulations. Modularity in assessment routine refers to modification of one rule not affecting the functionality of other rule. In the assessment routine, we can observe that there is no dependency between RULE statements. Every rule statement is independent of each other. Hence, the assessment routine supports modularity in the code which makes it flexible enough to cope up with the changes in the regulations.

Extensibility Extensibility here refers to the ability to add new rules to the regulations easily. The assessment routine presented in the thesis supports re-usability of code. That is, New rules can added by just reusing the RULE specification in the code. This makes the assessment routine extensible. Figure 5.10 depicts the compliance results of z/OS system against modified regulation which consists of 20 rules. This shows that rules can be easily added to the regulation.

Compliance costs: Generally, time taken to check the compliance for a single system may take several days and include many people. The main objective of the thesis is to reduce the compliance costs and time taken to check the compliance per system. The assessment routine presented in the thesis takes 0.35 seconds as shown in Figure 5.9 to check the compliance of a single system against a regulation, which consists of six rules or the controls. Figure 5.11 shows that time taken to check the compliance of a z/OS system against regulation after adding sixteen new rules remains same i.e., 0.35 seconds. This shows, even when the number of rules in a regulation increases, the time taken to check the compliance will be not significant.

Complexity: The complexity here refers to the complexity of modelling a rule or control in a compliance checking language. The already existing methodologies [28], [17], [19] use a very complex compliance checking languages which makes it difficult to formulate a control/rule. The CARLa statements STANDARD, DOMAIN, and RULE used in the assessment routine makes it very easy for one to define a regulation/standard, define domains and Rules. A rule specification consist of a set of test specification. A TEST specification is used to formulate a compliance requirement which consists of a test variable which is the field name of a particular resource, relational operator to give the compliance condition such as equal to, less than, and greater than etc., and compliant value specified in the regulation. Using this test specification, compliant requirements can be easily formulated. The combination of all the above statements makes it easy for one to formulate a control/rule.

Compliance progress traceability: Compliance management is a continuous process. In every iteration of compliance management, the organization tries to improve upon the compliance status of a system. Hence, the compliance report or compliance results generated in every iteration should be able to present the compliance progress achieved

in the current iteration. This helps in tracking the compliance progress of every iteration. The assessment routine used to check compliance in this thesis allows tracking the compliance progress of a system by presenting *compliance number* in compliance results where compliance number is the total number of compliant objects in a domain.

Chapter 6

Conclusions and Future work

The main goal of the thesis was to propose to a methodology to automate the process of compliance checking for z/OS against multiple regulations across multiple domains. This was accomplished by following the approach presented in Chapter 1. A secondary goal of the theis was to identify and address the limitations facing by the current approaches and products.

The primary goal of the thesis was achieved by proposing a methodology for a general automated compliance management in Chapter 4. This consisted of mainly 6 processes, Regulation analysis, Control Extraction and Classification, Test Compliance, Reporting, Deviation analysis, and Deviation handling. Each of the processes were explained in detail.

The thesis presented a brief description of mainframe operating system, z/OS and its external security manager, RACF which is used to monitor and collect all the security related information in z/OS. We also discussed about IBM security zSecure component structure, IBM security audit in specific which enables analysis and reporting on mainframe security events. It is also used for compliance reporting for the z/OS environment.

Thesis discusses about the consequences of non compliance and problems associated with the implementation of automated compliance checking facing by the already existing approaches in chapters 4 and 1 respectively. Thesis also provides clear definitions of compliance management, compliance checking, regulation, and security controls.

In chapter 5, we discussed about implementation of automated compliance checking of z/OS based on the proposed methodology. A brief introduction to the regulations, DISA STIG and GSD 331 is given. Also, examples of security controls of these regulations were presented to give a glimpse of security control structure. The security control families that these regulations addresses are also listed in the thesis. An event/action

based methodology for classifying the security controls of these regulation is discussed. Based on this methodology, the controls were classified to determine the percentage of automation of the controls of both GSD 331 and DISA STIG for z/OS. There are totally 339 controls from both the regulations out of which 41% of controls can be fully automated for z/OS, 20% of controls cannot be automated, 28% of controls can be automated in future for z/OS, and 9% controls can be partially automated for z/OS.

The secondary goal of the thesis was achieved by identifying and addressing the limitations facing by current compliance management approaches and products. The limitations were identified in Chapter 1. The limitations include: Complexity, complexity here refer to the complexity of formulating a rule, inability to support multiple types of regulations in compliance checking, inability to check compliance across multiple domains, huge compliance cost and time needed to check the compliance of a system, inability to support flexibility in the design, inability to support extensibility in the design, and inability to track compliance progress of a system. These limitations were addressed by implementing an efficient assessment routine presented in Chapter 5. Section 5.4 discusses about how the assessment routine addresses these limitations. The assessment routine was written in CARLa. Chapter 5 gives a brief introduction to this language. In order to implement the assessment routine, we extended CARLa by adding a new CARLa statement called STANDARD, and newlist type=compliance to perform meta reporting of the results of testing data from other newlist types.

The assessment routine was made as simple as possible to simplify the task of formulating rules/controls which solves the problem of complexity of compliance checking. The assessment routine consists of DOMAIN statement through which various domains to which the rules apply can be defined. This shows that assessment routine supports various domains in compliance checking.

The assessment routine supports multiple regulations of different types such as HIPAA, PCI DSS, DISA STIG etc., by using CARLa members. Each CARLa member holds the carla program for a single regulation. So, in order to check compliance against new regulations, new CARLa members can be added to the CARLa library. These members are added as input to the program by using IMBED statement in JCL. The program is executed in batch mode to check compliance with multiple regulations.

The assessment routine supports modularity in the code, i.e., modification to one rule does not affect the functionality of other rules. This makes the assessment routine flexible to cope up with the changes to the regulations. The assessment routine also supports suppressing few rules from compliance checking by using SUPRESS statement in the routine. The routine is also extensible in terms of number of rules. It supports re-usability of the code that helps to add new rules to the routine easily.

In general, time taken to check the compliance for a single system may take several days and include many people. The assessment routine presented in the thesis takes

0.35 seconds as shown in Figure 5.9 to check the compliance of a single system against a regulation, which consists of six rules or the controls. Figure 5.11 shows that time taken to check the compliance of z/OS system after adding sixteen new rules is still 0.35 seconds. This shows, even when the number of rules in a regulation increases, the time taken to check the compliance will be not significant.

The compliance results are designed to display the compliance number, actual value, compliant value, rule name, regulation name, rule/control description, and total number of both controls with which the system is compliant and non-compliant. The intent of reporting this information in the results is to simplify the task of deviation analysis and deviation handling in compliance management. The compliance number conveys the total number of both compliant and non-complaint objects of a domain in the system. This helps in keeping track of compliance progress.

The limitation of the current approach is that the assessment routine developed in the thesis is environment specific. The environment here is z/OS a mainframe operating system. Another limitation of the project is that it is difficult to check the compliance of a system with the rules which are too general. For example, few rules of PCIDSS are not specific to z/OS and hence, difficult to automate for z/OS. In order to automate the controls fully, it requires the rules to be specific for the environment.

6.1 Future work

In this thesis, automated compliance checking of z/OS is done against two regulations, GSD 331 and DISA STIG. As a next step, we aim to extend the compliance checking of z/OS against other regulations such as HIPAA, PCI DSS, and ISO 27002. We also aim to determine the percentage of automation of the controls existing in these regulations for z/OS. The analysis of DISA STIG and GSD 331 shows that only 40% of the controls can be automated fully, 20% of the controls can be automated in the future, and 9% of the controls can be automated partially. Future work aims to increase the number of controls that can be automated fully for z/OS by proposing ways to implement partially automated controls fully and implementing the controls that can be automated in the future.

The future work also involves extending the compliance checking to other environments such as z/VM. The compliance results generated in the thesis can used to generate the automated compliance report using Security Content Automation Protocol (SCAP). Future work also involves researching automated report generation using SCAP. One of the main challenges in compliance checking is to implement an efficient compliance checking process. Organizations never wish to spend much time in compliance checking process. They prefer having an automated compliance checking tool which takes less time. Also, the requirements of organizations with regard to compliance checking is

never ending. So, future work involves exploring ways to meet these requirements.

References

[1] *Mainframe Concepts*, IBM Redbook, 2008, http://publib.boulder.ibm.com/infocenter/zos/basics/topic/com.ibm.zos.zmainframe/zmainframe_book.pdf.

[2] Mike Ebbers, John Kettner, Wayne O'Brien, Bill Ogden, emphInroduction to the new Mainframe z/OS Basics, IBM Redbook, third edition, March 2011.

[3] Paul de Graaff, Ted Anderson, Julie Bergh, Peter Desforge, Lynn Kearney, Lori Halberts Kikuchi, Toni Nix, Mark Shell, *CA-Top Secret to OS/390 Security Server Migration Guide*, IBM Redbook, First Edition, October 2000, http://w3.itso.ibm.com/itsoapps/redbooks.nsf/RedbookAbstracts/sg245677.html?OpenDocument

[4] IBM, *z/OS Security Server RACF System Programmer's Guide*, IBM Redbook, SA22-7681-12, 2010

[5] IBM, *Security Server RACF Command Language Reference*, IBM Redbooks, Seventh Edition, 2004.

[6] IBM, *Security Server RACF Security Administrator's Guide*, IBM Redbook, SA22-7683-14, 2011.

[7] Axel Buecker, MichaelCairns, Monique Conway, Mark S.Hahn, Deborah McLemore, Jemie Pease, Lili Xie, *IBM z/OS Mainframe Security and Audit Management Using the IBM Security zSecure Suite*, IBM Redbook, Second Edition, August 2011, http://www.redbooks.ibm.com/redbooks/pdfs/sg247633.pdf

[8] IBM, *Information Security Controls for Customer XYZ*, Version 6.01, June 28, 2011, http://w3.ibm.com/bluepedia/display/en/GSD331

[9] DOD, *z/OS RACF STIG*, Version 6, Release 9, Oct 28, 2011, http://iase.disa.mil/stigs/

[10] PCI Security Standards Council, *Payment Card Industry (PCI) Data Security Standard, Requirements and Security Assessment Procedures*, Version 2.0, October 2010,

https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf

[11] Department of Health and Human Services, *HIPAA Security Series, Security standards: Administrative Safeguards*, Volume 2, 2005, revision: 2007, http://www.hhs.gov/ocr/privacy/hipaa/administrative/securityrule/adminsafeguards.pdf

[12] Department of Health and Human Services, *HIPAA Security Series, Security standards: Technical Safeguards*, Volume 2, 2005, revision: 2007, http://www.hhs.gov/ocr/privacy/hipaa/administrative/securityrule/techsafeguards.pdf

[13] Department of Health and Human Services, *HIPAA Security Series, Security standards: Physical Safeguards*, Volume 2, 2005, revision: 2007, http://www.hhs.gov/ocr/privacy/hipaa/administrative/securityrule/physsafeguards.pdf

[14] ISO/IEC, *Information technology-Security techniques-Code of practice for information security management*, First Edition, 2005-06-15, http://isvoc.com/download/ISO_27002_EN.pdf

[15] Liu, Y., Muller, S., and Xu, K. *A static compliance-checking framework for business process models*, IBM Systems Journal 46, 2 (2007), 335-361.

[16] Ghanavati, S., Amyot, D., and Peyton. L, *Towards a framework for tracking legal compliance in healthcare*,Proceedings of the 19th international conference on Advanced information systems engineering, Springer-Verlag (2007), 218232.

[17] Marwane El Kharbili, Sebastian Stein, Ivan Markovic, Elke Pulvermiller, *Towards a framework for Semantic Business Process Compliance Management*, Proceedings of GRCIS 2008. Volume 339, pages 1 - 15, Montepellier, France, June 2008. CEUR Workshop Proceedings.

[18] El Kharbili, M., Stein, S., and Pulvermller, E, *Policy-Based Semantic Compliance Checking for Business Process Management*, Proceedings of MOBIS Workshops MOBIS Conference, Saarbrcken, November 2008, (2008), 178-192.

[19] Yip, F., Parameswaran, N., and Ray, P, *Rules and Ontology in Compliance Management*,11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), (2007), 435-435.

[20] IBM, *Select the right solution to help simplify the challenges of compliance management*, Last Accessed: March 2007, ftp://public.dhe.ibm.com/software/tivoli/buyers-guides/compl_mgmt_gde.pdf.

[21] Krishna Sapkota, Arantza Aldea, David A Duce, Muhammad Younas, Rene Banares-Alcantara, *Towards Semantic Methodologies for Automatic Regulatory Compliance Support*, PIKM11 Proceedings of the 4th workshop on workshop for

Ph.D. students in information & knowledge managemen, 2011, 83-86.

[22] Eudralex, *The Rules Governing Medicinal Products in the European Union*, Online, 2011. http://ec.europa.eu/health/documents/eudralex/cd/index_en.htm

[23] Sesen, M.B., Suresh, P., Banares-Alcantara, R., and Venkatasubramanian, V, *An ontological framework for automated regulatory compliance in pharmaceutical manufacturing*, Computers & Chemical Engineering 34, 7 (2010), 1155-1169.

[24] Sumner Blount, CA Security Management, *The Role of Security Management in Achieving Continuous Compliance*, October 2006.

[25] R.L. Graham, D.E. Knuth, and O. Patashnik, *Concrete mathematics*, Addison-Wesley, Reading, MA, 1989.

[26] James E. Purcell, *Security Control Types and Operational Security.*

[27] Marwane El Kharbili, *Business Process Regulatory Compliance Management Solution Frameworks: A Comparative Evaluation* 8th Asia-Pacific Conferencxe on Conceptual Modelling (APPCCM 2012), Melbourne, Australia, Conferences in Research and Practice in Information Technology (CRPIT), Vol. 130, January-February 2012.

[28] Marwane El Kharbili, Qin Ma, Pierre Kelsen, and Elke Pulvermueller, *CoReL: Policy-Based and Model-Driven Regulatory Compliance Management*, EDOC'11, Preceedings of the 2011 IEEE 15th International Enterprise Distributed Object Computing Conference, 2011, 247-256.

[29] National Institute of Standards and Technology, *Information Security*, Publication 800-53, Revision 3, .2009

[30] http://www.emc.com/security/rsa-archer/rsa-archer-compliance-management.htm

[31] http://www.mycompliancemanager.com/solutions/compliancemgmt.aspx

[32] http://technet.microsoft.com/en-us/solutionaccelerators/cc835245.aspx

[33] http://accelus.thomsonreuters.com/solutions/compliance-management/accelus-compliance-manager

[34] http://www.soxlaw.com/

[35] http://www.ca.com/~/media/Files/ProductBriefs/compliance-mgr-ps.pdf

[36] Michael Kibort,*Achieving PCI Compliance on the Mainframe*, April 2011.

Appendix A

CARLa Fields of newlist type "Compliance"

1. STANDARD: Name of the standard. The field length is 24.

2. STANDARD_DESC: Description of the standard. Maximum length is 255, default width is 48.

3. STANDARD_VERSION: Version of standard. The field length is 24.

4. RULE: Name of the rule. The field length is 64.

5. RULE_DESC: Description of the rule. Maximum length is 255, default width is 64.

6. TEST_NEWLIST_TYPE: The newlist type where this test is checked.

7. DOMAIN_SELECT: The set of ¡type¿(selclause) clauses that defines the domain selection. This field is just included to ease detailed reporting and make it unnecessary to refer back to the STANDARD definition. The default width is 64, the maximum length is 32767.

8. DOMAIN_SUMMARY: The set of ¡type¿(keyvars) clauses that defines the domain summarization

9. DOMAIN_NAME: A name for the domain to help understanding. The field length is 32.

10. RULE_EXEMPT_SELECT: A selection clause for exempt objects that need to be counted separately in a rule-level summary report, this field is just included to

ease detailed reporting and make it unnecessary to refer back to the STANDARD definition. The default width is 64, the maximum length is 32767.

11. RULE_EXEMPT: A flag indicating that this object while in the domain is not to be counted as a compliance failure if that occurs. The TEST_COMPLIANT still will show whether the exempt object is compliant or not.

12. COMPLEX: Complex of system tested if a security database rule or setting is involved. For non-complex-specific tests, this is missing. The default width is 8, maximum length 13.

13. SYSTEM: System tested. For complex-level or non-complex-specific tests, this is missing. The length of this field is 8.

14. CLASS: SAF class of the resource name identifying the object tested. For non-resource-specific tests, this field is missing. The length of this field is 8.

15. RESOURCE: Resource name identifying the object tested. For non-resource-specific tests, this field is missing. Max length is 246, default 44.

16. RESOURCE_LOCATION: Location where the resource exists. See description of this field in other places.

17. VOLSER_KEY: Volume serial that also is part of the object key. This field is missing if the volume serial is not needed to uniquely identify the object.

18. SENSITIVITY: Sensitivity of the resource, if any. The length of this field is 11.

19. AUDITCONCERN: Risk associated with the resource, if any was defined for the resource (built-in explanation). This is not part of the standard, but may further explain the reason for the standard rule.

20. AUDITPRIORITY: Priority associated with the audit concern, if any was defined for the resource. This is not part of the standard. It is present to help prioritize non-compliant findings.

21. RACF_CLASS: Class of the RACF profiles contributing to the object protection being verified.

22. RACF_PROFILE: RACF profiles contributing to the object protection being verified. For non-RACF systems or non-profile rules the field is missing. The profile exists in the database implied by COMPLEX.

23. ACF2_RULEENTRY: ACF2 rule entry of the object protection being verified. For non-ACF2 systems or non-rule-entry rules of the standard, the field is missing.

The rule entry exists in the database implied by COMPLEX.

24. TEST_NAME: Name of the test. The length of this field is 24.

25. TEST_FIELD: Specification of the field to be tested. This may be a lookup specification. For instance, a userid in the newlist type being of a certain privilege type can be tested through a lookup. For the base field of the lookup see TEST_FIELD_BASE_VALUE. The maximum length of this field is 255, the default width is 24.

26. TEST_FIELD_BASE_VALUE: This shows the value of the base field in a TEST_FIELD lookup specification, for instance USERID in TYPE=TRUSTED.

27. TEST_FIELD_VALUE: Value of the field. The maximum length of this field is 4096. The default width is 48.

28. TEST_RELOPER: Relational operator for test. Field-to-field compares are not supported here. The width of this field is 2.

29. TEST_COMPLIANT_VALUE: Value the field is compared against according to the relational operator TEST_RELOPER. The maximum length of this field is 4096. The default width is 48.

30. TEST_COMPLIANT: Flag field that indicates whether the field value tested was OK. Note that a false value does not necessarily mean non-compliancy. That is only the case if field TEST_NONCOMPLIANT is true.

31. TEST_NONCOMPLIANT: Flag field that indicates whether the field value tested was OK. Note that a false value does not necessarily mean compliancy. That is only the case if field TEST_COMPLIANT is true.

32. SUPPRESS: Flag indicating rule suppression.

33. SUPPRESS_REASON: Text string with reason for rule suppression. The maximum length of this field is 255. The default width is 48.