BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2022 Bc. Patrik Končitý

BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS
ÚSTAV TELEKOMUNIKACÍ

USER RIGHTS DEFINED BY SELINUX TECHNOLOGY IN RED HAT ENTERPRISE LINUX OPERATING SYSTEM
TITLE OF STUDENT'S THESIS

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. Patrik Končitý

ADVISOR
VEDOUCÍ PRÁCE

prof. Ing. Dan Komosný, CSc.

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study program **Information Security**

Department of Telecommunications

Student: Bc. Patrik Končitý ***ID:*** 203172

Year of study: 2 ***Academic year:*** 2021/22

TITLE OF THESIS:

User rights defined by SELinux technology in Red Hat Enterprise Linux operating system

INSTRUCTION:

Study possible options for user's privilege restriction in the OS Red Hat Enterprise Linux using the SELinux technology (Security-Enhanced Linux). Propose general rules based on the actions of typical OS users. Implement the rights proposed into the SELinux technology. Verify their operation using case studies.

RECOMMENDED LITERATURE:

[1] VERMEULEN, S. SELinux Cookbook. Packt Publishing, 2014, 240 s. ISBN 978-1783989669.

[2] Linux Dokumentační projekt. 4. vyd. Computer Press, 2008. 1336 s. ISBN: 978-80-251-1525-1.

Date of project specification: 7.2.2022 ***Deadline for submission:*** 24.5.2022

Supervisor: prof. Ing. Dan Komosný, Ph.D.

Consultant: Mgr. Zdeněk Pytela

doc. Ing. Jan Hajný, Ph.D.
Chair of study program board

ABSTRACT

This thesis deals with the restriction of user rights in the Red Hat Enterprise Linux (GNU/Linux) operating system using SELinux technology. The first part of the thesis explains the mechanisms by which we can restrict rights in the OS. The basic two mechanisms of rights restriction are explained. These mechanisms are Discreet Access Control and Mandatory Access Control. Next, we elaborate on SELinux technology, which is a form of Mandatory Access Control. In the practical part, first a survey of typical OS users and their typical activities is carried out, followed by the implementation part where the theoretical knowledge of SELinux technology is applied. The implementation of new confined users and the templates that are added to each security policy are described. These security policies are then tested on the configured systems to test the mitigation of specific vulnerabilities. Mitigation is successful and the attacker is prevented from gaining privileges. In the last section, we discuss possible future extensions to the security policies.

KEYWORDS

Discretionary Access Control, Mandatory Access Control, Linux, Red Hat Enterprise Linux, Security, SELinux Technology, Policy, Confined Users

ABSTRAKT

Tato diplomová práce se zabývá omezením uživatelských práv v operačních systému Red Hat Enterprise Linux (GNU/Linux) pomocí technologie SELinux. V první části práce jsou objasněny mechanismy, pomocí kterých můžeme omezovat práva v OS. Jsou zde vysvětleny základní dva mechanismy omezení práv. Jedná se o mechanismy Diskretního řízení přístupu a Mandatorního řízení přístupu. Dále rozvádíme SELinux technologii, která je formou mandatorního řízení přístupu. V praktické části je proveden nejdříve průzkum typických uživatelů OS a jejich typických činností, na které navazuje implementační část, kde jsou uplatněny teoretické poznatky o SELinux technologii. Je zde popsána implementace nových confined uživatelů a šablony, které jsou přidávány do jednotlivých bezpečnostních politik. Následně jsou tyto bezpečnostní politiky testovány na konfigurovaných systémech a testuje se mitigace konkrétních zranitelností. Mitigace je úspěšná a útočníkovi je zamezeno získat práva. V poslední části se zaobíráme možnými budoucími rozšířeními bezpečnostních politik.

KLÍČOVÁ SLOVA

Diskrétní řízení přístupu, Mandatorní řízení přístupu, Linux, Red Hat Enterprise Linux, Zabezpečení, Technologie SELinux, Bezpečnostní politiky, Omezení uživatelé

ROZŠÍŘENÝ ABSTRAKT

Tato práce se zaměřuje na bezpečnost linuxových systémů z obecného hlediska a zejména na technologii SELinux. Útoky na počítačové systémy jsou v každodenním životě na vzestupu a je třeba se před těmito hrozbami chránit. Jen za rok 2020 bylo v produktech firmy Red Hat nahlášeno více než 3 000 bezpečnostních problémů.[1]

K zabránění těchto útoků nebo případné zmírnění škod v dnešní době naštěstí existuje několik bezpečnostních mechanismů, které poměrně účinně chrání. Pro operační systémy GNU/Linux existuje několik typů mechanismů, které dokáží mitigovat případný úspěšný útok.

Většina linuxových systémů používá systém diskrétního řízení přístupu. Je to prostředek omezení přístupu k objektům na základě identity subjektů nebo skupin, ke kterým patří. Subjekt s určitým přístupovým oprávněním může toto oprávnění předat jakémukoli jinému subjektu. [2] Mechanismus, založený na diskrétním řízení přístupu, je však pro zabezpečení systému nedostatečný. Povolení nebo odepření přístupu je založeno pouze na identitě a vlastnictví uživatele a nezohledňuje další relevantní informace z hlediska bezpečnosti. A vzhledem k nedostatečnému zabezpečení DAC byl zaveden mechanismus, založený na povinném řízení přístupu (MAC), který má zlepšit nedostatky mechanismu DAC. MAC, na rozdíl od DAC, vynucuje administrativně nastavenou bezpečnostní politiku pro všechny procesy, soubory v systému a všechny subjekty, přičemž rozhodnutí jsou založena na štítcích obsahujících různé informace důležité z hlediska bezpečnosti. [3]

Tento mechanismus je implementován v systémech Red Hat Enterprise Linux a nazývá se Security Enchanced Linux (SELinux), který zvyšuje bezpečnost systému a zmírňuje dopady útoků hackerů. Pomocí systému SELinux můžeme nejen zmírnit dopad přímých útoků, ale také díky možnosti omezit konkrétní uživatele můžeme snížit dopad útoků sociálního inženýrství. SELinux aktivně nezabrání útočníkovi útok na zařízení, ale do velké míry zmírní dopad škod.

V teoretické části je nejdříve popsán mechanismus diskrétního řízení přístupu (DAC), který se nachází na valné většině operačních systému GNU/Linux. Jsou popsány základní entity, které mohou přistupovat k objektům a také jaké základní akce mohou provést na objektech. Nevynechali jsme ani speciální práva které mohou být na objektech.

Po prvním bezpečnostním mechanismu pokračujeme s popisem mandatorního řízení přístupu (MAC). Vysvětlujeme jak funguje linuxový bezpečnostní modul (LSM), který je základním kamenem celého bezpečnostního mechanismu a jakým způsobem posuzuje jednotlivé žádosti objektů o přístup ke zdrojům. Následuje kapitola 1.2 o SELinux technologii, která popisuje historii tohoto bezpečnostního mechanismu a základní principy SELinuxu. V následujících podkapitolách je popsán princip jakým způsobem funguje SELinux bezpečnostní server na systému a jak posuzuje

přístupy subjektů k objektu.

Popisujeme princip bezpečnostního kontextu, kdy jakýkoliv subjekt i objekt na systému má bezpečnostní kontext, a na základě pravidel má subjekt povolený přístup do bezpečnostních domén. Dále jsou popsány další SELinuxové prvky a jsou uvedeny typy bezpečnostních politik a stavy jejich vymáhání na systému. Můžeme mít teoreticky tři stavy bezpečnostních politik na systému.

SELinux mechanismus může být úplně vypnutý, kdy nefunguje a bezpečnostní politiky se nevymáhají. Pak existuje permisivní mód, kdy politiky nejsou vymáhány, ale všechny nevyžádané přístupy subjektů k objektům jsou uloženy do logů. A ještě může nastat třetí stav SELinux politik, kdy bezpečnostní server aktivně vymáhá všechny nevyžádané přístupy.

Bezpečnostní politiky se skládají z pravidel, a struktura obecného pravidla je vysvětlena v teoretické části diplomové práce. Můžeme mít několik typů pravidel: od základního pravidla jež povoluje přístup subjektu ke zdrojům, až po pravidlo, které povoluje původnímu procesu začít proces potomka s jiným bezpečnostním kontextem.

V poslední části teorie popisujeme jakým způsobem dokáže SELinux omezit přímo GNU/Linux uživatele a mitigovat tak eskalaci práv. Jsou zde vypsány existující uživatelé, které můžeme najít v SELinux mechanismu a rozsah jejich práv. V další části je probráno jakým způsobem mohou existující omezení uživatelé rozšířit své pravomoce a jaké jsou limity stávajících uživatelů. Tímto končí teoretická práce, která je takovým základním průřezem bezpečnostních mechanismů, které jsou k dispozici na OS GNU/Linux.

Jako hlavní cíl praktické části diplomové práce bylo navrhnout a vytvořit nové omezené SELinuxové uživatele, kteří budou pro systémové správce snazší na vytváření a údržbu. V první kapitole je proveden průzkum typických činností uživatelů OS GNU/Linuxu aby bylo možno určit, která funkcionalita bude pro různé typy uživatelů. Na základě průzkumu bylo určeno jaké typy uživatelů budeme vytvářet a jakou funkcionalitu budeme v politikách pokrývat.

Je rozhodnuto, že budou tři typy uživatelů. Prvním uživatelem bude uživatel basic_u, který bude pokrývat základní funkcionalitu jako například přihlášení do systému přes konzoli, možnost spustit systém s GUI nebo například možnost pracovat se základními příkazy v terminálu a schopnost pracovat s webovým prohlížečem. Toto není ovšem celý výčet funkcionalit, více v kapitole 2.3.3.

Poté tu máme uživatele advanced_u který má hlavní výhodu v tom, že může využívat nástroj sudo pro spouštění některých příkazů jako root, ovšem tento typ uživatele nemůže přes sudo spustit všechny programy, má povolenou jen základní funkcionalitu jako například správa služeb pomocí systemd. Další funkcionalitou který má tento uživatel přidanou pomocí šablony na systému je možnost pracovat

jako čtenář se SELinuxem. To znamená, že uživatel může vidět aktivní pravidla, vlastnosti SELinux mechanismu nebo například aktivní booleany.

Třetí typ uživatele admin_u má nejširší rozsah pravomocí. Má možnost ovládat administrátorský software, který vyžaduje spoustu práv jako například cron, wireshark nebo například měnit parametry kernelu. Také může pracovat s všemožnými logy a spravovat ostatní uživatele na systému. Dále má práva pracovat se systémem SELinux a má právo SELinux deaktivovat.

Poté jsme zvolili vhodný systém, na kterém budeme moci nové uživatele vytvářet, a nakonfigurujeme systém pro potřeby vývoje nových uživatelů. To znamená stáhnout software, který slouží k vývoji bezpečnostních politik a jejich udržbě.

Tyto tři uživatele jsme se tedy rozhodli vytvořit a jejich politiky se budou skládat z variabilních šablon, které vytvoříme. Tyto šablony se budou dát snadno použít a přidat do politik, bez nutné hlubší znalosti SELinux mechanismu.

V implementační části popisujeme jakým způsobem lze vytvořit omezeného SELinuxového uživatele na systému. Také dále popisujeme všechny vytvořené šablony, které použijeme v bezpečnostních politikách. Detailně rozebíráme jaké pravidla je do nich potřeba přidat, aby splňovala definovanou funkcionalitu. Dále také popisujeme samotnou konfiguraci nových omezených uživatelů na systému a funkčnost nových bezpečnostních politik.

V předposlední části zkoušíme na našem systému s nakonfigurovanými uživateli dva exploity na eskalaci práv a pozorujeme, jakým způsobem může SELinux mitigovat tyto útoky. Naši SELinuxoví uživatele dokáží tyto eskalace práv úspěšně mitigovat a nedovolí útočníkovi získat na systému větší rozsah práv. V poslední kapitole diskutujeme o možných vylepšeních do budoucna jako například přepínání do jiných uživatelských rolí v bezpečnostním kontextu.

Ve zkratce tedy jsou vytvořeny šablony na základě typických činností uživatele OS, které lze variabilně použít, a v budoucnu může být pro systémového správce snazší pracovat se SELinux mechanismem a vytvářet si vlastní SELinuxové uživatele, či použít námi vytvořené uživatele.

Výstup diplomové práce je dostupný veřejně na internetu, pomocí `gitového` repozitáře, který obsahuje šablony a vytvořené vzorové uživatele. Je zde také uveden návod jakým způsobem na systému nakonfigurovat tyto SELinuxové omezené uživatele.

Šablony jsou otestovány na námi konfigurovaném systému, a vyzkoušeli jsme si jak obstojí proti eskalacím práv na systému. Díky mandatornímu řízení přístupu, které běží nad klasickým diskretním řízením přístupu se podařilo SELinuxu absolutně eliminovat eskalaci práv na OS. Funkčnost práce byla tak důkladně prověřena. V průběhu vývoje šablon se nám navíc podařilo objevit několik chyb v aktivních SELinuxových politikách a mohli jsme je tak nahlásit společnosti Red Hat, která je

odstranila.

KONČITÝ, Patrik. *User rights defined by SELinux technology in Red Hat Enterprise Linux operating system*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications, 2022, 67 p. Master's Thesis. Advised by prof. Ing. Dan Komosný, CSc.

Author's Declaration

Author:	Bc. Patrik Končitý
Author's ID:	203172
Paper type:	Master's Thesis
Academic year:	2021/22
Topic:	User rights defined by SELinux technology in Red Hat Enterprise Linux operating system

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank my supervisor prof. Ing. Dan Komosný Ph.D. for his professional guidance, consultation, patience and suggestive suggestions for the work. Special thanks to colleague from Red Hat, Mgr. Zdenek Zpytela, Senior Software Engineer, for his excellent guidance and collaboration in sharing his broad knowledge and experience in SELinux Technology. Moreover, I want to express my sincerest thanks to all my colleagues from the Security Controls team. I would also like to thank my colleague Bc. Vladimír Janout for providing me with stimulating ideas related to priviledge escalations.

Contents

List of Figures

List of Tables

Listings

Introduction

This thesis focuses on security of linux systems in general sight of view and especially based on technology SELinux. Attacks on operating systems are on the rise in everyday life and we need to protect ourselves from these threats. Often these attacks are caused by some vulnerabilities in the code that are caused by incorrectly written code.

For 2020 alone, more than 3,000 security issues have been reported on Red Hat products.[4] And some security tools are needed to mitigate these vulnerabilities as much as possible when they are discovered and not yet patched. For linux operating systems, there are several types of security to mitigate the effects of weaknesses in case an attacker gains control of the machine.

Most linux's use a Discretionary Access Control system, which control how subjects interact with other objects, and how subjects interact with each other.[5] However, a mechanism based on discrete access control is insufficient for strong system security, allowing or denying access is based only on user identity and ownership, and does not take into account other relevant information from a security perspective. And due to the inadequate security of DAC, a mechanism based on mandatory access control (MAC) was implemented to improve the inadequacy of the DAC mechanism. A MAC architecture, unlike a DAC, needs the ability to enforce an administratively set security policy for all processes and files in the system, with decisions based on labels containing various security-relevant information. [6]

This mechanism has been implemented in Red Hat Enterprise Linux systems and is called Security Enchanced Linux (SELinux), which makes system more secure and mitigate impacts of hackers attacks. With SELinux, we can not only mitigate the impact of direct attacks, but also with the ability to restrict specific users, we can reduce impact of social engineering attacks. The need to restrict specific users has recently increased due to these attacks. It is by restricting users using SELinux that the damage to the system can be mitigated.

These SELinux confined users will be major topic of this diploma thesis. We will focus more on these users and discuss the features and capabilities of this part of SELinux technology. As the one of practical objectives of the thesis will be to conduct research on typical GNU/Linux user activities and to define several types of constrained users on this basis. However, the main goal of this work is to create security policies that will restrict GNU/Linux users on the system, and thus have the power to better mitigate privilege escalation on the operating system.

The diploma thesis is divided into a theoretical part and a practical part. The chapter 1.1 will first discuss the security mechanisms used by GNU/Linux operating systems and discuss the principle of each mechanism. The general principles of these

mechanisms and their basic properties will be described.

In the next chapter 1.2, we will discuss the specific SELinux security technology that this thesis will cover. In the first related subchapter 1.2.1 we will discuss the basic features of this technology, describe the mechanism of operation and the structure of the rules used in SELinux. In the second subchapter 1.2.2, we will look at how we can use SELinux technology to define privileges for users of the operating system. We will describe the current implementation of confined users in SELinux technology.

The practical part of the thesis starts with chapter 2.1. In this chapter a survey of typical users of GNU/Linux operating systems is conducted. The possible classification of users into groups based on system usage is described and then the possible typical deployments of each type of user are discussed. In the next chapter 2.2, based on the survey of typical OS users, a possible new architecture for constrained SELinux users is proposed.

The next chapter 2.3 will describe the implementation of the security policies that will be used for new confined users, and will provide instructions on how to create your own confined user and link it to a classic GNU/Linux user. It will also describe the templates we will use to build security policies for individual SELinux users. The penultimate chapter 2.4 will show how SELinux can mitigate privilege escalation on the operating system in our configuration. And the last short chapter 2.5 will outline possible future enhancements to the security policies.

1 GNU/Linux systems and security

1.1 Security mechanism in GNU/Linux systems

In GNU/Linux-based operating systems, there are several ways to secure impacts from potential attacks, and this paper will focus on one of them, which will be discussed later in this paper. Operating system security is always an important topic because in times of increasing attacks, each additional layer of security can reduce the impact of an attack or directly prevent a given attack.

In this paper, we specifically focus on security mechanisms in systems that directly mitigate the impact of attacks. This is called proactive security and we have several elements that can help us achieve it.

The first security feature of these systems, is the existence of a special user called **root**, who has administrative privileges on the system. Unlike Windows systems where anyone can be an admin. So root has almost complete rights to the system and can do anything and anywhere. At the same time, when a user gets root permissions, root permissions can be divided into permission levels called capability, which allow only a certain range of root privileges. And this ensures even greater distribution of rights in the event of a root attack.[7] There are other users who do not have such rights in the system and can only perform common tasks and operations.

Separation of users and distribution of privileges makes it difficult for malware to spread throughout the system and take control of it. This model of rights distribution is called Discretionary Access Control (DAC) and will be described more in the chapter 1.1.1.

This model works on the basis of an access list, where only specific users are allowed access to certain objects.

The disadvantage of this solution is the lack of working with other important security information that can help to make the system more granular, but this is not the only way to prevent damage. Another option is to use a solution for operating systems added to the kernel as a module that offers greater granularity and can therefore specify the rights of users or applications on the system.

We call this solution "Mandatory access control" (MAC) and like DAC it is also a type of access control, but unlike DAC, MAC security policy is controlled centrally by the administrator who is tasked with this. The user does not have the ability to modify their security rights. For example, to grant himself the right to access a file that he should not otherwise be able to access.[8] In chapter 1.1.2 we will discuss this area in more detail and describe the MAC principle.

1.1.1 Discretionary Access Control

Discretionary Access Control (DAC) is a type of access control defined by the Trusted Computer System Evaluation Criteria. [9] It is based on access control by verifying to which group or entity an object belongs. The owner of the file or the user who has ownership rights is able to manipulate the object and can set access rights for other users and groups based on this.

Thus, access rights to an object can be granted to three entities:

- owner,
- group,
- other.

The **owner** is usually the owner of the file or a specific user account that is allowed access. A **group** is usually a group of users, or a specified group of users. And the entity **other** assigns rights to any process that does not correspond to any user or group.

We can allow several actions to each entity that handles the object. We can allow it to:

- read - **r**,
- write - **w**,
- execute - **x**.

These permissions are implemented as permission bits attached to each file and can only be set by the owner of that object.[10]

Thus, each entity can be assigned one of these rights in any combination. It can have all of these rights or even none. Unless an entity is assigned a right it is unable to override that right. The only super entity that is able to bypass these rights is root, which was mentioned in chapter 1.1. This simple form of access control list has been further supplemented with additional special rights to the object.

Setting permissions on an object in this way ensures that when any user runs this file, they take over the user ID of the owner or group of this executable. The owner who sets these special rights must be careful how they are handled, as this is a potential security risk. For example, if a special permission is set on a file that has the root ID on it, a user who runs that file can take over the root permission.[11] We have several types of special authorisations:

- setuid - **s** or **S**,
- setgid - **s** or **S**,
- Sticky Bit - **t** or **T**.

For each process that accesses an executable file with the **setuid** permission, access is granted based on the owner of the file, not the user who runs the file. With this special permission, users can access files and directories that are accessible only

to the owner. The **setgid** permission is a similar special permission to setuid, but unlike setuid, the user runs the executable with group permissions. If we want to apply the setuid or setuid permission to a file without enabling the execute bit, this permission is written with a capital **S**. If we want to enable the execute bit at the same time, this permission is written with a lowercase **s**.

The last special permission we can apply to an object is the **Sticky bit** permission. The sticky bit can be applied to any file or directory on the system. This permission prohibits the user from deleting other users objects. Again, we have the option to apply this permission to the object with or without the execute bit. The lower case **t** represents the Sticky bit permission with the execute bit enabled and the upper case **T** represents the sticky bit permission with the execute bit disabled.[12] In modern GNU/Linux systems, the sticky bit is used only for directories.[13]

DAC is the basic access control mechanism in Linux systems. It is relatively simple and suitable for use on systems that do not handle sensitive data. Discretionary access control fulfills two functions. An entity that does not have the right to modify an object is not able to modify that object by writing or otherwise. Furthermore, a subject who does not have the right to read an object is not able to read it. Thus, discrete access control can ensure the integrity and confidentiality of data on the operating system. On the other hand, for systems working with sensitive data, mechanisms with more granularity are needed. A possible security risk to the machine's security is the root account. If an attacker compromises the account or a process with its privileges, he can take control of the entire system and it's data.[14] However, root privileges can be limited and divided into capability that offers only a certain slice of root privileges. Still, a user who gets root privileges gets very strong privileges for operations on the system. Therefore, the root account needs to be restricted as much as possible, and this can be done with a security concept that will be discussed in the next chapter.

1.1.2 Mandatory access control

Mandatory access control (MAC) is a type of access control that gives a subject the right to perform an action on an object based on a security policy. Compliance with security policies is mandatory for MAC, and security policies can be set by the system owner and are usually implemented by the system administrator or security administrator. Even users with root privileges cannot override these policies if they are active, unless they have it enabled in the policy. MAC makes file and process protection independent of the object owner.[15]

LSM mechanism

Mandatory Access Control (MAC) uses the Linux Security Module (LSM) mechanism. This generic framework has been added to the kernel since version 2.6. Based on this concept, a module implementing MAC can be added to the Linux kernel. There are several types of MAC-based security modules and the system administrator can choose any implementation.[16]

The mechanism for using the module is as follows: an application or process makes a system call to the kernel with a specific request for access to an object or resource. The call first goes through error checking and DAC checking, in each of these checks the request may be denied and not granted.

However, if the system call successfully passes all these parts, the kernel calls the LSM hook, a module that extends the kernel behavior, and asks the module whether to allow or deny access to the object or resources based on active security policies. If the LSM module finds a rule in the policy allowing this access it will allow the action and if not it will deny it. This mechanism is shown graphically in the figure below:[17]

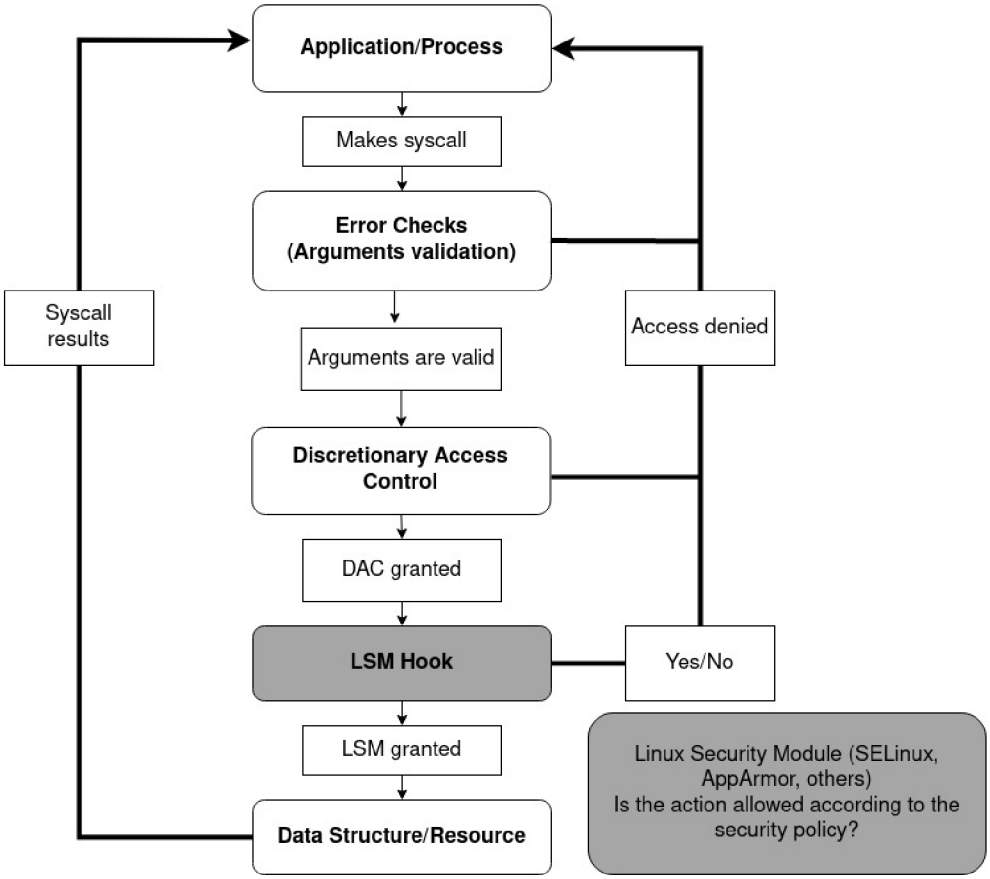

Fig. 1.1: Linux Security Modules (LSM) hooks

We have several different implementations of MAC technology that use the LSM module, the most well known of which are AppArmor and SELinux. In this paper we will look at SELinux technology in more detail, describing its mechanisms and looking at how this technology can be used to restrict users on a GNU/Linux system.[18]

1.2 SELinux technology

The history of **SELinux** dates back to the 1990s when the National Security Agency (NSA) and other organizations began developing a tool for comprehensive and flexible management of mandated access control. These organizations jointly developed a prototype of SELinux called the Flux Advanced Security Kernel (FLASK). FLASK was an implementation of a series of utilities that enforced mandated access control on a given system. Later, Red Hat expressed interest in a complete implementation of FLASK in its commercial Red Hat Enterprise Linux (RHEL) distribution, and SELinux was born. SELinux subsequently spread to other GNU/Linux distributions such as Debian GNU/Linux, Gentoo Linux and Fedora. [19]

Before we get to the more details about SELinux, let's summarize a few basic features:

1. When SELinux is active on a system, SELinux policies use rules to allow certain entities to access certain objects and allow various operations on them (e.g. open, read, write).
2. SELinux can restrict a service in its own "domain", thereby limiting its privileges to a level sufficient to do its job. If the service will require access to other resources and objects. This access must be allowed in the SELinux policy.
3. If the service wanted to perform something that is forbidden in the active policy, SELinux would not allow this operation.
4. If the service wanted to do something that is not explicitly allowed in the active policy, SELinux would not allow this operation.
5. GNU/Linux users can be mapped to SELinux users, and the permissions that a specific user needs on the system can be set for each specific user.
6. SELinux is not capable of stopping all malware that infects a system, however it can mitigate the effects of an attack.
7. New rules can easily be added to SELinux policies using tools such as audit2allow and others. [20]

1.2.1 Basic SELinux mechanism

First, let's discuss the basic elements of the SELinux module. Below is a diagram of the SELinux components that manage and enforce policies and contains all of the following parts:

1. An entity that requires an action on an object, such as reading the object.
2. An administrator of an object who knows the actions from the resource and can enable these actions if allowed by the security server. Sends queries to the security server.
3. A security server that decides the right of an entity to perform an action on a given object based on the rules in the policy.
4. A policy in which rules are written using a special policy language.
5. A cache in which the security server's decisions are stored.

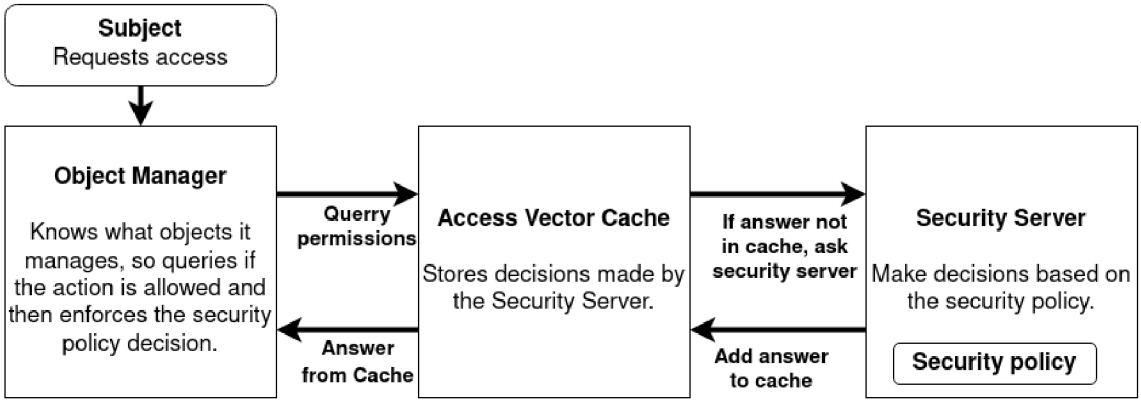

Fig. 1.2: High Level Core SELinux Component

The security server searches the active rules and tries to find the rule for the subject's access to the object. If it does not find the rule, access to the object is denied. The server that makes this decision needs to know the security context of the subject and object in order to make the correct decision.

SELinux context

The SElinux system context is a fundamental element of the policy, every file, directory, process or any system resource has a context. Based on this context, rules can be created in various forms and decisions can be made about how each entity accesses objects. Thus, every element of the system must always have a context. For example, objects for which do not have clearly defined contexts using special types of rules called file transition, the object obtains the general context of the parent directory.

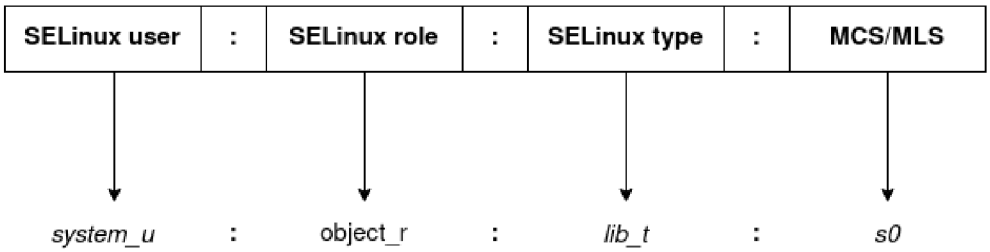

Fig. 1.3: Security context

SELinux User

This section is suffixed with `_u` such as `system_u`. This is the identity of the GNU/Linux user in the SELinux mechanism. We will discuss users in more detail in chapter 1.2.2.[21]

SELinux Role

Role is with suffix `_r` such as `object_r`. A SELinux user can have more than one role, and each role can have different permissions. A role can be associated with multiple possible types that allow permissions. We will discuss roles in more detail in chapter 1.2.2.[22]

SELinux type

The type part is with suffix `_t` such as `lib_t`. A type can be associated with either a process or an object. If the type is associated with a process, the type defines what the SELinux user's process is allowed to access. If the type is associated with an object, in this case the type defines what access rights the SELinux user has to that object. The type could be considered the most important part of the security context; the type is used to shape the access rules between subjects and objects.[23]

MCS/MLS

This part is only active if the SELinux policy is also activated with MCS or MLS. It consists of a sensitivity level or range entry to the security context. In the default policy state, the sensitivity is set to s0. Although neither MLS nor MCS is the main topic of the diploma thesis, we will talk more about MLS in next subsection.[24]

Two forms of MAC in SELinux:

SELinux can support two MAC types based on which the security module decides whether to allow or deny access.

- **Type Enforcement** - This MAC works on the principle of everything is denied by default. The default state is policies in which no rules are defined. If access needs to be granted, then the appropriate rule must be created. For example, if a process wants access to object, it needs to have that access defined by rules. In practice, this is the most commonly used mandatory access control form for SELinux.
- **MLS** - This type of MAC security mechanism is based on the Bell-La Padula (BLP) model, which is used for companies that need to split access levels on a system due to classified information. This security mechanism ensures that elements in the system at the same sensitivity level can only read and write at that level. Anyone further up can report to higher levels, but only the same and lower levels can read. [25]

Form of SELinux rules

This subsection explains what a common SELinux policy rule looks like, based on which the security server makes decisions. The basic structure of a rule is as follows:

rule_name source_type target_type:class perm_set;

- **rule_name** - This part of the rule determines the final nature of the rule, the allow, dontaudit, auditallow or neverallow action can be applied.[26]
- **source_type** - The label of the entity that wants to allow or disallow access to something is inserted here. It can be a process that wants to access the object.[27]
- **target_type** - Usually the label of the object that the entity wants to access and perform some action on it, such as reading, writing, etc. [28]
- **class** - This is the object class that the subject wants to access. This can be a class such as file,dir etc. There are also classes for different kinds of sockets.[29]
- **perm_set** - In this part of the rule, we fill in the type of action we want to allow on the object, such as reading, mapping, writing, and more.[30]

In a policy, we can have several types of rules, we can have classic permission rules that allow access to the target domain file. We can also have transition rules that provide either domain transition or object transition.

A domain transition rule ensures that a process in a particular domain can create a process that has a different domain and a different security context. In the case of object transition, this is when the new object requires a different label than it's parent. These transition rules are also important rules that we can add to policies. [31]

Type of active policies

We have several types of policies that can be active on the system and enforce access control.

- **Minimum** - This type of policy emphasizes the smallest possible policy size. Only a small number of services are restricted, usually services that are critical to the system. The rest of the services and objects and other system elements run as unconfined. This saves space on the operating system. [32]
- **Targeted** - A common type of policy on Red Hat systems. This policy covers a large number of services and other features on the system. By default, however, the user runs with the unconfined label, but it is possible to restrict the user. The MAC mechanism is in type enforcement mode.[33]
- **MLS - multi level security** - This type uses type enforcement mode and also MLS mode.[34]

Modes of SELinux policy

If the SELinux module is enabled, it can run in two modes on the system:

- **Enforcing** - security policies are active and actively enforce access,
- **Permissive** - security policies are active but do not enforce access, only logging denied accesses.

This was the chapter that explained the basic SELinux mechanisms, how the SELinux module works, the SELinux policy parts, the basic form of SELinux rules and more. In the next chapter, we will cover how to restrict GNU/Linux users using SELinux.

1.2.2 User rights defined by SELinux

The main topic of this paper is user constraints using SELinux technology. In this chapter we will describe what methods we have for restricting users using SELinux and also mention the current implementation. In the chapter 1.2.1, we have already written about how to label a GNU/Linux user, but here we will discuss it in more detail.

Each user of the system should be mapped to a SELinux user. Every element in the system must have a label, and so must the user. So even a newly created

user has a label. Only one SELinux user can be mapped to a GNU/Linux user, but multiple users can have the same SELinux user. By default, unless otherwise set in the system, each user is mapped to the default SELinux user **unconfined_u**. However, this default label allows almost unlimited privileges on the system, and if this user is compromised by an attacker, it can be difficult to mitigate the effects of an attack.[35]

Confined users

Fortunately, we can map GNU/Linux users to more restrictive SELinux users. In addition to unconfined_u, we have the following types of SELinux users:[36]

- **guest_u**,
- **root**,
- **staff_u**,
- **sysadm_u**,
- **system_u**,
- **user_u**,
- **xguest_u**.

Each of these types of users has different permissions and powers. The basic scope of permissions and authority for all users except unconfined and root is shown below:

User	Role	Domain	X Window System	su or sudo	Networking
sysadm_u	sysadm_r	sysadm_t	yes	su and sudo	yes
staff_u	staff_r	staff_t	yes	only sudo	yes
user_u	user_r	user_t	yes	no	yes
guest_u	guest_r	guest_t	no	no	no
xguest_u	xguest_r	xguest_t	yes	no	Firefox only

Tab. 1.1: SELinux users and their rights

The table does not include root or unconfined users because they have unlimited privileges.

Role-Based Access Control

Each of these types of users has different permissions and powers. Sometimes a situation arises where a confined user needs to extend their own permissions, and one

of the easiest ways to do this is to switch to a different role if they are allowed to do so.

Roles in SELinux technology work based on the Role-Based Access Control (RBAC) security model. Each user can have one or more roles defined to which they can switch and extend their privileges. A table of all users and their ability to switch to other roles is below:[37]

	sysadm_r	staff_r	user_r	guest_r	xguest_r	unconfined_r
sysadm_u	X					
staff_u	X	X				X
user_u			X			
guest_u				X		
xguest_u					X	
unconfined_u						X
root	X	X				X

Tab. 1.2: SELinux users and their switching

In SELinux there is an implementation of confined users and switching to other roles, but this implementation is outdated and does not meet the needs of today. A lot of services that are for restricted SELinux users do not exist anymore or have extended functionality and not all accesses are allowed for them. Also, the eventual intervention in the policies and reworking of the confined users policies for the needs of the system administrator is difficult in terms of the complexity of modifying the policies and finding enabling rules.

However, the main drawback of the current confined users architecture is the architecture itself, which operates on the inverted pyramid principle. That is, the user at the bottom of the pyramid has the fewest privileges, and gradually other users higher up the pyramid take over all of his privileges and extend them. However, they can take over rights that they don't inherently need.

Therefore, the goal of this work is to change the architecture of constrained users to meet current needs. This is because the original concept was created twenty years ago and does not suit today's operating systems, which often have different usage patterns and the users who operate them also have different habits of using operating systems. As far as threats are concerned, the world has also moved on in twenty years and threats are often more sophisticated and target different vulnerabilities on the system.

Thus, in this work, the focus is on the issue of confined users, who through reimplementation will restrict privileges on the system to the right extent to mitigate the impact of attacks. Selected services that are really used by real OS users will be treated for new confined users. At the same time, creating custom security policies should be easier for system administrators managing the system due to the modularity of the architecture. In the capabilities of this work, it is not possible to cover all the uses of services on the system, but the focus will be on covering the frequently used services by a typical user.

2 Practical part

This chapter will describe the practical part of this thesis, research will be conducted on typical uses and deployments of GNU/Linux users. Based on this, new users in the new architecture will be proposed. The implementation section first describes how to create a custom SELinux restricted user on GNU/Linux systems, then describes the individual templates that allow individual users to access resources and data on the system.

Then, the functionality of configuring the system with restricted users using the created templates is described, and the last subsection of the implementation paper discusses the possibility of mitigating privilege escalations on the system by using our new users with new security policies.

2.1 Survey on typical GNU/Linux users

In this part of the thesis, a survey of typical GNU/Linux deployments and users will be conducted. The aim will be to identify the types of users based on their use of the services and features of the system. Based on the result of the survey, we will be able to propose a redesign of limited users in SELinux technology. As the main source of data for the survey, we will use articles on the Internet that describe the usage of GNU/Linux systems from several perspectives.

Linux Documentation website, which provides documentation for GNU/Linux operating systems, can also serve as a possible source of inspiration.[38] We will also draw on the original SELinux restricted user types and the official SELinux policies. Another source may be the ref policy, these are policies that are an alternative to the Red Hat supported SELinux policy, some things in the ref policy are different from the official policy.

Types of users using GNU/Linux

As far as user types are concerned, we have several sources to find out what types of users the systems are available for. The first source is to look at the different distributions and what versions each distribution is offered in. For example, the fedora distribution is officially offered in two versions. The fedora workstation version and the fedora server version. This offers the basic idea that you will need to divide the users based on the capabilities and usage of the system into the ordinary user who just wants to use the GNU/Linux system as a working tool for their purposes and the user who has more knowledge and uses the system as a server.[39]

Another source may be the already active SELinux policies that divide and then restrict users into several types. The division and restriction of users under official SELinux policy was discussed in the chapter 1.2.2. In ref policy, the distribution of confined users is virtually identical. There are a large number of user types in these policies, and the exact use and deployment of these SELinux users is not specified.

So basically, the user is using OS for normal use or is using GNU/Linux in a server version and there is a need for increased powers.

Activities of typical OS users

In this chapter, we describe the basic activities of typical OS users. We've already established that there are two broad groups of users-ordinary system users and system administrators. Now we will need to conduct a survey to determine the typical activities on the system when it is used by an ordinary user and when it is used by a system administrator.

With this, we will find out what to focus on in the new policies and add that to the new policies. Most of the things that will be enabled for the normal user will be enabled for the system administrator. The first part that needs to be enabled and there is no need to do research on it for typical usage is logging into the system itself using the console, then remote login, then GUI login, there are other ways to log in, but due to the scope of a thesis only these login methods will be addressed.[40]

At the same time, not only the GUI startup should be handled, but also the subsequent system setup in the GUI. This could be another module that would be available in the new policy for new limited users. Also, the user should be able to use the file explorer that is freely available on the system if the system runs with the GUI.

Moving on to working with the terminal, the user should be able to use basic file handling commands such as copying files, listing files, searching through files and so on.[41] Also, an ordinary user should be able to use the text editor in the terminal such as vim or nano.

However, the key for today's ordinary OS user is the use of the Internet. Most users no longer work with the system as such but take advantage of the Internet and use the Software as a Service (SaaS) principle. The typical ordinary user needs to work with a browser to connect to the software he wants to work with.[42] We also need to ensure that the user is able to download files from the browser and also that the user is able to, for example, start a video with sound in the browser. The last activity that should be enabled for the ordinary user is working with USB and other peripherals that can be connected.

This section described how ordinary users use the GNU/Linux operating system. The next section discusses typical use of the Linux operating system from the perspective of a system administrator.

A system administrator should basically be able to do the same things as a normal user, which means of course logging into the system in different ways, being able to work with the file explorer if the system runs with a GUI, or being able to use basic terminal commands.

The system administrator should still control the more advanced utilities that are related to configuring network connections and displaying network information. Examples of such utilities include ping, netstat, nslookup, and others. [43] We do not cover all types of utilities in this survey, but only the most common or well-known ones. Covering all types of uses would be beyond the scope of this paper. The system user should also be able to work with user management, i.e. create new users, delete them and change them.

Another activity that an administrator should be allowed to do, and it is often the administrator's job, so he should be able to work with cron and should be able to work with server logs, that is, collect them and work with them. Also, as a user who works as an administrator, he should be able to work with software installation, either using package manager or manually. The system administrator should be allowed to work with tools that monitor properties about the server, such as server uptime or memory usage. [44] Also, working with SELinux security policies should be enabled. Other typical admin use cases would be beyond the scope of the thesis.

Typical activities that users should have enabled and that are most commonly used are discussed in more detail below:

Basic checks

- user logs in with a password and can execute a command,
- systemctl user@ID status,
- systemctl status –user,
- journalctl, journalctl –user,
- user can log out successfully, no error message pops up,

Basic operations

- invoke a terminal (GUI),
- date, ls, ps, man,
- systemctl –user, journalctl –user,
- password setup, update, with a password policy in place,
- internet access,
- poweoff, reboot,

Advanced operations

- systemd user service and timer unit, transient unit,

- systemd services control commands - resolvectl, hostnamectl,
- ssh/scp , changing configuration in /.ssh,

Operations with files

- browse, read, edit, delete, copy,
- /tmp access,
- /homedir access,
- /run access,
- /etc access,
- /usr/share access,
- ~/.local, ~/.cache and ~/.config can be updated,
- using find,
- using locate,
- executing files in ~/bin, /usr/local/bin,
- executing /tmp,

Devel work

- shell and python scripting,
- gcc,
- make,
- make install,
- git, git config,
- using screen/tmux,

GUI

- start, login, autologin,
- limits.conf applied,
- manage stuff in gnome settings,
- images viewer,
- camera,
- pdf viewer,
- screensaver locks and unlocks with a password,

Browser usage

- browse, http, https, ftp, configuration,
- web browsing,
- download and store, download and open, upload,
- video and music replay, editing,
- browser plugins,

Office tools

- word processor,
- sheets,
- presentation,

Peripherals usage

- kbd, mouse, USB drive, other drives, bluetooth, wireless mouse, headphones, thunderbolt,
- printer,
- scanner,
- camera,

Network operations

- address management - static, dynamic,
- virtual private network,
- ping,
- ifconfig,
- cli tools for setup network - ip, nm,
- firewall,
- cli tools for troubleshooting - route, netstat, traceroute,
- ping,
- ipsec,

Admin tools

- users management,
- authconfig,
- passwd,
- vipw,
- sudo,
- systemd settings (e. g. RemoveIPC),
- cron,
- wireshark,
- logging,
- sysctl tunables,
- tuned,

Security

- getsebool,
- checks - sesearch, seinfo, sestatus,
- newrole,
- ssh with another role,
- auditing (ausearch),
- semanage,
- chcon,
- semodule disable, enable, load,
- restorecon.

This chapter has described the typical users of the operating systems and also

the typical use of the system from their point of view. Based on this list, we will create security policies for users and try to allow programs and tools from this list in security policies.

This list covers only the most commonly used tools and programs and does not take into account many other less commonly used tools. However, there is room for further improvements and enabling rights for lesser used tools in the future.

2.2 SELinux proposal

This chapter describes a possible new architecture limited to SELinux users. First, however, it is important to mention what the current restricted user architecture looks like in SELinux technology.

The current architecture works on the principle of an inverted pyramid, where the user with the fewest rights is at the bottom of the pyramid and users higher up the pyramid contain the rights of the lower user as well as additional rights. However, this architecture has one drawback, users that have more rights and take over the rights of lower users also take over rights that they do not necessarily need to use.

Another disadvantage that is not directly related to the current architecture is the addition of rules to the policies and the creation of the policies themselves, if a user wanted to allow the use of network utilities for example, they do not have a macro in the policy that contains all the necessary rules for the use of network utilities. For a more advanced user who wants to manage policies and add rules to them, this makes it difficult to know the rules or functions to add to active policies.

Design of new SELinux user architecture

The new architecture will work on a different principle, unlike the inverted pyramid, the range of rights for users is independent of each other. Each user has a range of rights tailored to their exact needs. Thus, there will be no cases where a confined SELinux user will have rights that he would not necessarily need for his typical activity. This would solve one problem with the current SELinux confined users architecture.

In the future architecture, the problem of complex policy creation will also be solved, special macros will be created, called templates. These templates will cover a set of functionalities. An example of a template is the template for user login via the console. With these templates, the administrator will be able to piece together policies for custom restricted users as desired or possibly supplement existing restricted

users with modules with needed functionality. This will make policy creation an easier task for someone who is not intimately familiar with SELinux technology.

Proposals of new users

In this section, we propose possible new users for the new architecture of confined users discussed in the previous subsection. The basic consideration is to create a confined user for the system administrator and another for the normal user.

It would require adding a limited ordinary user who could use the sudo utility. With the sudo utility, a user is able to perform operations with the permissions of another user, usually a root user. This user would be somewhere between an admin user and a regular user with a range of privileges. This user with more extensive rights would be possible as a default user in RHEL and Fedora operating systems instead of the unrestricted user that is in the current architecture. These three confined users would serve as the base SELinux users of the new architecture.

The basic user on the system will be the **basic_t** user, this user will be able to connect to the system via the console and will also be able to connect via the GUI. Once connected to the system, the basic user will be able to perform simple actions in terms of verifying the correctness of the system boot and all the threads related to the successful running of the system.

Furthermore, the user will also be able to see the system log and will be able to perform basic user settings such as changing the user password, changing the name and other settings. From the basic terminal commands, it will have the power to work in allowed file directories. He should be able to work with basic network utilities. It should also be able to work with system peripherals whether it is a mouse, keyboard or attached flash drives.

A basic user should also be allowed to control basic SELinux mechanisms such as looking up rules on the system or seeing a list of all booleans on the system and checking which attributes are assigned to which SELinux types. In the case of accessing the system through the GUI, the user has the right to change the settings of the desktop system. Last but not least, the user should be able to perform classic office work using office software and should also be allowed to work with a web browser.A basic user should also be able to work with development tools such as building a package or working with git. However, he won't be able to push commits via ssh because ssh will only be enabled for the next user.

This brings us to the next user advanced_t. This user will have the same privileges as the basic user but in addition to the previous user, he will be allowed to use ssh and scp. He will also be able to use the sudo utility if the user is in the sudo group. This will give him the power to configure and set up daemons via systemctl,

and he will be able to install packages using package managers such as yum or dnf.

The last user will be the admin_t user. This user will also be able to work with security on the system as far as SELinux modules are concerned, he will be able to add new rules and change the security context of files, he will also be able to disable the whole SELinux mechanism, he will have unlimited possibilities as far as controlling the SELinux mechanism is concerned. Another key power for admin users will be the ability to manage users on the system and set sysctl and kernel tunables. He will also be able to use various tools that require root privileges such as wireshark, cron or work with firewalls. It will also be able to work with logs.

These are the three types of users that we will be creating in this thesis and that will be the output of this thesis. We'll have a basic_u user that will serve as a user that could be on ordinary Linux workstations, then we have an advanced_u user that offers more privileges for developers working on the system, and then we have an admin_u user that will have unlimited privileges on the system. These three users will be the basis of user security policies on the system. The next chapter will describe the implementation of the new user security policies.

2.3 Implementation

First of all, it is necessary to clarify what will be the content of the practical output. The result of the practical output will be the creation of a new SELinux user for whom large macros will be created, which we will call templates because they will contain many rules and will always be related to some functional part of the system. It will be a set of rules for the functionality of the system. Specifically, the output of the thesis will be to create a template for logging into the text console. The reason for choosing this template is that in the new architecture, every user will have this template built into the policy. However, before that we have to choose what system we will accept it in and what tools we will use on the system.

Choice of system

The choice of the OS on which to develop the policies for new users was relatively easy. A lot of the credit for the development of SELinux technology goes to Red Hat, and in addition, SELinux is set up by default in Red Hat Linux distributions. Red Hat Enterprise Linux (RHEL) version 9.0 was chosen as the system on which to develop new users and modular policies for users.

The other option was to use the community Linux distribution Fedora, which also has an active SELinux module by default, but due to less stability of the system

the choice fell on the official Red Hat Linux distribution. The beta version of RHEL-9.0 is freely available on the official Red Hat website, the only condition is to register here.[45]

Choice of packages

In order to create policies and users, in addition to the SELinux module, it is also necessary to install several packages that help to work with policies, debug denials and create local security policies.

For a clearer listing of SELinux denials from the log, the **policycoreutils-python-utils** package is useful. This package includes the **audit2allow** tool, which translates the denials dump into an understandable policy dump. We can find out which package this tool is in by using:

Listing 2.1: How find source package for tool.

```
# rpm -qf `which audit2allow`
policycoreutils-python-utils-3.3-1.el9.noarch
```

Another useful tool found in this package is the **semanage** tool, which can manage various SELinux policy settings, and can map users to SELinux users.

Another package that has been installed is the **setools-console** package, which contains a useful tool called **seinfo** that can browse policies and output security context types and attribute associations. Another tool in this package is the **sesearch tool**, which searches for rules in an active policy. Next useful package is **policycoreutils**. In this package are tools such as **semodule**, which is used to load the local policy module into SELinux policy.

But the most important package to install is **selinux-policy-devel**. This is a package for developing local policies, and this package is central to that work. Together with the installation of the selinux-policy-devel package, the packages mentioned above are also installed as dependencies.

This was the section on the packages needed to develop new policies, and now on to the section on how to create a policy.

Creating a new user

This section will describe how to create a new user and map it to a new SELinux user. With this tutorial, you should be able to create a new SELinux user with a custom security context that gets a new context for login. To find out the user's security context after login, use:

Listing 2.2: How to get security context of the user.

```
# id -Z
```

If all steps are done successfully the context should look like this:

Listing 2.3: How look security context on system.

```
CustomContext_u:CustomContext_r:CustomContext_t:s0
```

If any part is set incorrectly, the system user will get the security context **unconfined_t**. With this tutorial you will create a custom SELinux user with basic settings, however to make a running system with this user functional you need to add additional rules to the policy to make all parts of the system work properly. The templates that will address the functionality of the new user on the system will be addressed in the next section of the thesis.

First, you need to create a local policy module for the user. It is therefore necessary to create the source files for the SELinux policy. The source files are as follows:

- **Type Enforcement (.TE) file** - This file contains the rules to be written for this user's type or domain transitions. Not only can rules be used here in raw form, but also in the form of macros that contain multiple rules and allow access to objects with a different type. What the rule looks like was described in chapter 1.2.1.
- **File Context (.FC)** - This file contains the security context of the files and folders associated with the domain.
- **Interface File (.IF)** - The interface file contains macros that allow manipulation of the domain. Macros consist of raw rules that allow various actions on the domain. Raw rules are the basic element of the policy, all macros and all other elements in the policy such as booleans are decomposed into basic raw rules. For example, a macro that allows access to the files of the policy module. This is important for other domains that need to access that domain's resources.

These three source files are then compiled to create a .pp module, which is then loaded into the SELinux policy. When creating a test user, the procedure will be displayed.

Our module will be named testUser and will have a **.TE** extension. It is used for testing purposes, so there is no need to use additional source files.

This file will look like this:

Listing 2.4: The form of the .TE file.

```
# cat testUser.te
    policy_module(testUser, 1.0)

    type testUser_t;
    role testUser_r;

    role testUser_r types testUser_t;
    allow system_r testUser_r;
```

For a SELinux user to function properly, several things need to be included in the user policy. First, the domain type and domain role need to be defined. Next, you need to link the role to the type and also enable the link between the system role and the domain role. These should be basic elements in any newly created policy and with this, the user will get the appropriate context defined in the domain when logging in. We have a source file with a basic role definition and a basic template for user login and now we will need to compile the source file as follows:

Listing 2.5: Building local policy.

```
# make -f /usr/share/selinux/devel/Makefile testUser.pp
Compiling targeted testUser module
Creating targeted testUser.pp policy package
rm tmp/testUser.mod tmp/testUser.mod.fc
```

A module is created and loaded as follows:

Listing 2.6: Loading module to kernel.

```
# semodule -i testUser.pp
```

It can then check if the module is loaded, or use the seinfo tool to check the newly created testUser_r role.

Listing 2.7: Testing if it's module loaded.

```
# semodule -l  | grep testUser
testUser
# seinfo -r testUser_r
Roles: 1
    testUser_r
```

Then you need to change the /etc/selinux/targeted/contexts/default_type file and add a new user testUser. The location of the file, depends on the active version of the policy, if the active version is targeted, the location in targeted would be as here in that case, if it was mls, the location of the file would be /etc/selinux/targeted/contexts/default_type. The file will look like this:

Listing 2.8: Adding new user type.

```
# cat /etc/selinux/targeted/contexts/default_type
auditadm_r:auditadm_t
secadm_r:secadm_t
sysadm_r:sysadm_t
staff_r:staff_t
unconfined_r:unconfined_t
user_r:user_t
testUser_r:testUser_t
```

You will still need to create a file for a new user in /etc/selinux/targeted/contexts/users, the easiest way is to copy an existing user and change the user name in the file using:

Listing 2.9: Adding new SELinux user.

```
# sed -e 's/user/testUser/g' user_u > testUser_u
# ls
guest_u  root  staff_u  testUser_u  unconfined_u  user_u
xguest_u
```

The last step is to map a GNU/Linux user to a SELinux user. First you need to create a new user for example newUser.

Listing 2.10: Creating new GNU/Linux user.

```
# useradd newUser
# passwd newUser
Changing password for user newUser.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

Then you need to create a new SELinux user based on the new type and role.

Listing 2.11: Adding role for SELinux user_u.

```
# semanage user -a testUser_u -R "testUser_r"
```

Now the user is linked to the SELinux user.

Listing 2.12: MappingGNU/Linux user to SELinux user_u.

```
# semanage login -a -s testUser_u newUser
```

And finally you need to change the labels in the home directory of newUser because it got the default security context.

Listing 2.13: Fix label of home directory for mapped new SELinux user.

```
# restorecon -RvF /home/newUser
```

After all these steps, the user newUser should get the security context of the new user testUser_u after logging in.

2.3.1 Developed policy

In previous part we showed how create a new user and now on to creating policies for the new user. It was mentioned at the beginning of the chapter 2.2 that the user will be made up of templates for the most part.

The functionality and number of templates was described in chapter 2.2. In the thesis, the first important templates was developed, namely the templates of user login to the text console. It still needed to be verified that the user actually works up after logging in without any problem and this can be verified using a command:

Listing 2.14: Status of running user.

```
$ systemctl status user@{UID}
```

The command will show us the systemd user manager for the running user and we can see if the user is running fine and if all services bound to the user have started.

New SELinux user

In the first step, a new SELinux user **basic_u** was created with the role **basic_r** defined. This user will serve us as a basic user for diploma thesis. The user was created using the tutorial on creating users in the 2.3 chapter. The SELinux user was then mapped to the GNU/Linux user basic_user.

However, we are not able to log into the console because the user does not have the other necessary privileges that will be addressed in the login template. However, if we change the policy state from enforcing to permissive using:

Listing 2.15: Setting mode of SELinux mechanism.

```
# setenforce 0
```

We are able to log into the console. This command can change the active policy state. Number 1 means switching to enforcing mode and number 0 means switching the policy to permissive mode. We discussed policy states in more detail in the chapter 1.2.1.

After all these steps, the basic_user user should have the following security context when logged into the console:

Listing 2.16: SElinux status of GNU/Linux user.

```
$ id -Z
basic_u:basic_r:basic_t:s0
```

Logging templates creation

In order to be able to log in via the console even in SELinux policy enforcement mode, you will need to create a template for logging in via the console. The basic policy that was defined in the basic.te file looked like this:

Listing 2.17: Essential policy for basic user.

```
# cat basic.te
policy_module(basic, 1.0)

type basic_t;
role basic_r;

role basic_r types basic_t;
```

It defines the type and role of this policy and the interrelationship between type and role. To make the policies as modular as possible, we will move these type and

interconnection definitions to an .if file where we will create a template for logging in via the console. The template will be named **confinedom_user_login_template** and will look like this:

Listing 2.18: Interface file for local basic policy.

```
# vi basic.if
template(‘confinedom_user_login_template’,‘

    type $1_t;
    role $1_r;

    role $1_r types $1_t;
    ’)
```

Instead of a fixed type and role, there are parameters in the macro into which the variable is inserted. Using the macro in basic.te will look like this:

Listing 2.19: Using first template in basic .TE file.

```
# cat basic.te
policy_module(basic, 1.0)

confinedom_user_login_template(basic)
```

This step will make policy creation for SELinux users very modular and allow easy creation. Many more rules need to be added to this macro to allow the user to log into the console. And you just add a template to the .te file and there is no need to add additional rules that specify the types and roles used in the policy and set up the connection between role and type. All these rules will not be described and shown in detail here, because it is a big bunch of rules, but it will be explained in general what all the SELinux user needs to enable in order to boot without problems. At the same time, the local policies will be freely available on `github` to test the implementation.

2.3.2 Templates

This chapter will describe the templates that will be used to enable rights on the system and that will be created for this work. A template will consist of several rules that will enable the functionality of a given tool or program on the system. Without adding these templates to an active SELinux restricted user policy, users will not be able to use the required functionality. SELinux policies work on the principle that everything is disabled by default and functionality on the system needs to

be enabled by adding rules to the security policy. There will be several of these templates covering specific areas of activity on the system. For example, there will be a template to allow limited user logins in the text console, a template to cover functionality to allow the sudo tool, or a template to allow the use of administrative tools such as the cron daemon or the use of wireshark.

The various sub-sections related to templates are not alphabetical but based on creation and level of authority, meaning that templates related to admin and security work will be at the end of this section.

Login template

A larger number of rules were needed to allow login to the console. Only the key parts that were needed will be described here. It was necessary to allow the new user to use the terminal, and it was also necessary to allow the user to run all binaries, since the user needs to run several utilities when logging in. Then it was necessary to allow the user to manage their own processes and set setgid and setuid on their own files. Next, we needed to enable socket communication for different socket types for the type that will use this template.

You also need to allow access to /etc/passwd, where the user can read the essential data that is needed during login.

Another set of rules that are important are the rules that allow systemd to work, because systemd is used to run other processes in userspace.

Furthermore, with this template, the user's security context can be added to the attributes that hold additional rules relevant to logging in via the console. One of these attributes is the userdomain attribute, which enables rules for all user domains, so there is no need to add these rules individually for each domain, but just add the domain to the attribute.

You need systemd to be running properly for the system to run properly. Systemd can then be used to start and manage other system services and daemons. Systemd can also have other systemd subprocesses that provide login, time synchronization or logging, for example. All of this also needs to be implemented in a template.[46]

You also need to enable the SELinux user's communication template with the kernel for login and enable reading kernel information. It was also necessary to enable logging related operations in the policy, as it is necessary to keep logs of events in case of any errors on the user side.

An important part of the console login template is a set of rules related to SELinux technology. These are rules that allow the display of information about SELinux,

such as what mode the SELinux technology is running in, the ability to read security contexts, or the ability to create security contexts, for example.

The last important part that needed to be enabled was the handling of the home directory and the user temporary directory. In the previous term paper there were problems with incorrect security contexts for the temporary directory, but fortunately this has been solved in the thesis and the user login is without problems.

SSH template

This template enables user login using the ssh communication protocol, as well as other communication protocol related operations such as connecting via ssh to various git repositories and more. This template also allows you to use scp, which communicates using the ssh protocol. This template also allows you to use scp, which communicates using the ssh protocol.

The basic idea was to add the already existing **ssh_role_template()** template that allows the domain to use the ssh protocol and also needed to enable read and write inherited sshd pty for the user domain. In addition, the domain needed to enable read and write operations to the pty multiplexor (/dev/ptmx). A dytransition from the sshd domain to the user domains was added to the template.

Basic commands template

Another template that has been worked on is the template for basic commands, which mostly dealt with basic commands that the user uses in the terminal for various operations. The user can check the systemd status of the running system to see if the user has logged in with no problem and all parts of the system are running successfully. In the case of the GUI, the user has the option to launch the terminal. He also has access to the system log. The user can monitor processes using the ps tool.

The image below shows a user in enforcing mode with the security context basic_u running successfully. This verifies the successful implementation of the console login template.

Listing 2.20: User running succesfully.

```
Last login: Mon Nov 22 22:34:41 2021
$ id -Z
basic_u:basic_r:basic_t:s0
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
$ systemctl status user@1002
user@1002.service - User Manager for UID 1002
Loaded: loaded (/usr/lib/systemd/system/user@.service;
static)
Active: green active (running) since Mon 2021-11-22
22:35:09 CET; 17s ago
Docs: man:user@.service(5)
Main PID: 1052 (systemd)
Status: "Startup finished in 59ms."
```

However, this is not the only functionality covered by this template, the user can browse directories using ls or has the power to open manual pages and can set the date and time on the system using the date tool.

There are also rules added to this template that allow the passwd tool to work, allowing the logged in user to change their password.

To enable these functionalities, it was necessary to enable the reading and mapping of systemd daemon files in the policy, which provides information about the running user on the system.

Reading of logs was enabled for the ability to view the user's journals, and mapping of manual files was also enabled for the ability to be read by the user.

Last but not least, access to date files needed to be enabled to change and display the time. To allow the user to change their password, a macro was added to allow the passwd tool to run.

The next rule under consideration to be added to the basic command template after the security policies are complete is a dontaudit rule to read all files. Because the SELinux server logs a lot of access vector cache messages to allow accesses and resources for the domain that are not needed for the functionality of the tool or software. With dontaudit rules, the SELinux server can stop logging these resource access records when the access is not allowed by a classic allow rule.

Networking template

This template is smaller than the previous one because some of the rules that are needed are enabled in the previous templates. The template should cover the network utilities that are used to configure and debug networks. Such as the ifconfig utility used to configure the network interface or the ping, traceroute, netstat and route commands.

It should also cover the ip addr and ip link utilities, which will replace ifconfig in the future. These tools are enabled for the user by adding the macros **netutils_run_ping_cond()** and **netutils_run_traceroute_cond()**.

Additional rules did not need to be added to the template due to the existence of rules in the previous templates and the coverage of all necessary rules in these two macros.

Graphical login template

The template that allows user login via GUI is one of the most extensive templates created. It should be noted at the outset that this template was developed for the gnome desktop and may not be fully functional in other GUIs.

Not only does it allow the user to log into the GUI, but it also allows the user to work with the entire gnome desktop environment. This means the ability to make various settings on the system such as adding and managing internet connections, working with bluetooth, various settings and customizations to the appearance of the system, and setting up peripherals that can be connected to the system such as a monitor, keyboard and mouse.

You also need to enable working with other settings such as printer and scanner settings or user management and last but not least sound settings. There are many more things in gnome settings, but only the most common and most used ones have been listed here. You also need to enable native gnome desktop applications like pdf viewer, image viewer or camera control in the template. Also, don't forget the ability to download applications using the gnome sofware center.

The gnome software center is the only way for users to install applications unless the user is a sudoer and has a template that enables sudo on the system.

In order for the gnome GUI to work properly, several resource and file accesses had to be enabled. Since there are many rules that are needed for the GUI to work properly, not all of them will be listed here, but only the most important rules will be described.

For the basic GUI start-up it was necessary to enable the use of x windows system, which allows the GUI to be started on the linux operating system. Next, it was necessary to use a domain transition from the user domain to the x windows system domain and back.

It was also necessary for the GUI process that will run in the user domain to enable the process attribute retrieval, as well as the information and ability to set resources for the process. One of the other enabled rules that were added to the template was to enable socket communication for the domain.

Another segments that needed to be enabled are the rules dedicated to devices connected to the system such as various peripherals and hard drives, video cards, video and so on. All the important files for the devices are in the **/dev/** directory folder. These files need to be read and written to for proper functionality of the entire environment. We also need to write to the attachable disks that we can connect to the system so that, for example, we can work with the flash drive and modify the content on it.

We also need to enable the rtkit daemon, which acts as a resource allocator for processes, and we also need to enable the fuse software, which provides users with the ability to create their own filesystems. For SELinux policies, you only need to enable GUI read-only security contexts on the system to allow SELinux security contexts in file explorer.

The domain also needs to enable the Advanced Linux Sound Architecture, which we call **"ALSA"** for short, which handles sound. For the user domain running the GUI, the desktop system needed to be enabled to communicate with the application using the dbus daemon that handles this communication. There are many applications and daemons that need to communicate with the user domain via dbus. Dbus is the software that provides communication between applications. Such as the bluetooth daemon or the firewall daemon. There were a lot of rules related to dbus communication in this template and there is no need to describe them in detail.

It is also necessary to enable work with the printer and work with the scanner. These are the most important rules that needed to be added to the GUI template in order for the SELinux restricted user to work properly with the gnome GUI.

This template is likely to be used for all types of our new limited users, as a large proportion of systems today run with a graphical user interface.

Mozilla browser template

The template allows a limited user to use a web browser and also allows adding plug-ins to the browser that can improve its functionality. It was decided that a template would be written to use the mozilla firefox web browser. Several things had to be enabled, such as adding a macro to the template to allow the user domain under which the browser will run to work with plugins, or allowing the user domain to communicate with mozilla using the dbus daemon. Also, mozzila_t domain was enabled to work with mozzila_t domain in the process area and file transition for mozzila firefox files was enabled.

These are the basic rules that will allow a confined SELinux user to use the mozilla firefox web browser. The user is able to download files through the browser and can play various multimedia on the browser and surf the internet.

Sudo template

The template that allows you to use sudo is already a template for users who will have more privileges and will be able to run the software as root. The plan is to use this template for advanced_u and admin_u users. This is just to allow running commands with sudo, and in addition there are elementary rules in this template that will allow the new security context username_sudo_t to access resources on the system. The moment the user enters a command in the sudo terminal, after authentication, the user's security context changes to the sudo format mentioned in the previous sentences and the new security context needs to access resources on the system.

So it was necessary to add an existing **sudo_role_template()** to the function that covers the rules that enable sudo control, this template has all the rules needed to run this tool. It is also important to enable the possible basic rules that are associated with using the sudo utility. For example, the ability to enable or disable a service using systemctl, or the ability to load kernel modules into the kernel, or the ability to use sudo in an ssh connection. All of these rules have been added to the sudo template permissions.

Security template basic

Thanks to this template, the user who will have it in the active policy should be able to work with the basic tools of the SELinux mechanism. A configured user will be able to get a list of all SELinux booleans that are in the policy, they should also be able to search for all SELinux rules on the system or be able to find out which domains belong to a certain attribute and vice versa.

This functionality is used in case of debugging problems that may occur on a given system due to SELinux. Specifically, the **sesearch** tool is used to search for SELinux rules on a system, the **getsebool** tool is used to list all SELinux booleans for a change, and the **seinfo** tool is used to list attributes and domains.

To enable these functionalities, it was important that the template contain rules that allow reading SELinux policy and the ability to load SELinux modules into the kernel. It is likely that other necessary rules for proper functionality are enabled in the base templates.

Security template advanced

If we wanted to allow more privileges for limited users to work with the SELinux mechanism, we would use this template, which offers advanced options to modify security policies. It allows users to work with security policies such as adding new rules to an active policy, disabling domain-specific security policies via modules, or creating new security contexts for files on the system.

This is a list of only the most well-known operations that can be performed with the SELinux modification mechanism, however there are other possible SELinux modifications that will be described below with specific tools.

The most important tool for which functionality needed to be covered is the **semanage** tool, which provides a wide range of modifications to the SELinux mechanism. With this tool, booleans can be turned on/off or, for example, SELinux security contexts can be assigned to Linux users. Another tool is **ausearch** which can browse all records of SELinux denials that appear on the system.

The **semodule** tool can be used to upload, enable and disable domain-specific security modules to the kernel. With **chcon** it is possible to change the security context of files and **restorecon** utility checks the security context of files based on the policy, if it does not match the policy then this security context is changed to the one officially set in the security policy.

In the advanced security template, users will also be allowed to set the SELinux mode of the entire mechanism thanks to the **setenforce** tool. If we want to enforce policies, we use this command and add option 1 after the command, if we want to disable policy enforcement we use option 0.

To enable the selected functionality described here, we needed to enable the ability to read audit logs for the user domain in the rules if we want to access SELinux denials using the ausearch tool. To enable the SELinux mode setting, there is a macro **selinux_set_enforce_mode()** directly that enables changing the SELinux mode. For the ability to change boolean values for the user, the macro **selinux_set_all_booleans()** has been added to the template.

For the semanage utility to work properly, you need to enable the communication of this utility via the dbus daemon with the user domain and also add a macro that enables the usage. You also need to allow the user domain to change security contexts on the system.

Admin commands template

This is the last template that was created for confined users in this work and is intended for users who will have almost unlimited rights on the system. This template covers the various tools used by the administrator in his daily work.

Such as managing users on the system, using various network management utilities such as firewall or working with logs or wireshark. Also, working with the tuned daemon that checks all connected peripherals has been enabled, and a user who has this template added in an active policy can also change sysctl and kernel tunables.

The **iptables_run()** macro needed to be enabled to work with the firewall, and the **files_unconfined()** macro was added to work with any file on the system. For wireshark to work properly, a larger set of rules needed to be enabled because wireshark works with different socket types and reads all incoming communication on the system, also the existing macro **wireshark_role()** was enabled.

With the addition of the **logging_admin()** interface, the user domain will be able to work and view logs. The **tuned_dbus_chat()** template allows the user to communicate with the tuned daemon, allowing the user to use this tool.

This was the last template that was implemented for the new limited users, all the necessary and important functionality was managed to be added to the templates for the users.

2.3.3 Configuration of system with confined users

This chapter describes which templates each user is composed of. The composition of templates gives users different functionality. First, the individual policies need to be compiled and loaded into the system using the semodule utility, then the appropriate changes need to be made to the SELinux settings and the Linux user needs to be mapped to the new SELinux user.

The procedure will not be described in detail here because the 2.3 chapter describes it step by step.

For example, after mapping the advanced_u user with the new security domain, you should see the following output in the terminal after entering the following command:

Listing 2.21: Security context of user.

```
$ id -Z
advanced_u:advanced_r:advanced_t:s0
```

For the purpose of this work, three users basic_u, advanced_u and admin_u have been created with different responsibilities. However, thanks to the newly created templates, the person who will create the policies does not have to use the users we have designed, but can compose his own user from the templates. The templates make building a custom user much more user-friendly. Now on to how the individual users are configured and what functionality they fulfill.

Basic_u

This is the basic user with the least amount of authority. This user has a template added in the policy for logging in via the console **confinedom_user_login_template()** and also logging in via the GUI **confinedom_user_login_template()**. With the **confined_ssh_connect_template()** template, he can use the ssh protocol and **confinedom_networking_template()** allows the restricted user to use network utilities.

There is then the **confined_use_basic_commands_template()** template to allow the use of basic commands and the **confinedom_mozilla_usage_template()** template that allows the use of the Mozilla Firefox web browser.

A user with this range of permissions is able to work on the system as a normal user who needs to work largely with the browser and GUI and does not need other administrative utilities.

Advanced_u

This restricted user contains all the templates that were described for the first user in the policy, but also has two additional templates added. In the active policy, it has the added template **confinedom_security_template_basic()**, which allows the user to browse the active SELinux rules on the system and browse other SELinux mechanism data.

There is also the addition of the **confinedom_sudo_template()** template, which allows a restricted user to use the sudo utility, as long as they are in the sudoers group. However, this template allows him to use the sudo utilities for which the rules are written, meaning that he is not automatically allowed to run everything

with sudo. The SELinux server will block these attempts unless he is allowed to run a specific utility with sudo.

This restricted user may be ideal for people who need to perform more complex operations on the system.

Admin_u

This is the last user that was created in this work. The user has almost unlimited powers on the system and is allowed a wide variety of tools. This type of user is suitable for people who are system administrators, as it offers them the widest functionality. Two templates are added to the policy, the **confinedom_security_template_advanced()** template provides the user with the ability to change SELinux rules and security contexts. It can also disable the entire SELinux mechanism, thus making it inoperable. The second template added to the admin policy is the **confinedom_admin_commands_template()** template, which allows the user to use various advanced admin tools.

These are the three basic users that were created in this work. The individual functionality could be extended, but due to the scope and complexity of the work, the most common functionality is brushed off. However, as mentioned once in this chapter, a developer can create and assemble their own limited user quite easily thanks to the templates created.

2.4 Application to improve security

In this part of the thesis we will describe possible privilege escalations that can occur on a linux system and we can see how SELinux can or cannot prevent and mitigate these attacks from the user side. RHEL-9.0 was used as the target system on which the latest privilege escalation exploits were tested. These exploits were tested first on a system where SELinux was disabled, then on a system where SELinux was active but users were not restricted in any way, and then on systems where users had the security contexts of the new restricted users which we created.

Five priviledge escalations were searched for in the last year and a half, but only two of them could be reproduced on RHEL-9.0. In the table below we attach all the vulnerabilities we were able to find with all the useful links.

The two priviledge escalations that we managed to reproduce are the first two in the table. These vulnerabilities were disclosed this year. These vulnerabilities are **Dirty Pipe** and **PwnKit**.

Codename	Affected component	Public disclosure	CVE
Dirty Pipe	Linux kernel	2022-03-07	CVE-2022-0847
PwnKit	Polkit - pkexec	2022-01-02	CVE-2021-4034
NotQuite0DayFriday	iSCSI - Linux kernel	2021-03-12	CVE-2021-27365
Sequoia	Linux kernel - file system layer	2021-07-20	CVE-2021-33909
Linux eBPF	Linux kernel - eBPF	2021-09-01	CVE-2021-3490

Tab. 2.1: Table of priviledge escalations

2.4.1 DirtyPipe

Thanks to a vulnerability that security researchers have dubbed "Dirty Pipe", users can gain root privileges on a given system through publicly available exploits, making it potentially dangerous for vulnerabilities that would only allow an attacker to gain ordinary user privileges. With this privilege escalation, he can take control of the entire machine. This vulnerability first appeared and affected Linux Kernel version 5.8 and later. Android devices are also affected by this vulnerability. [47]

How DirtyPipe vulnerability work

Now to the vulnerability itself. Thanks to dirty pipe, an unprivileged user can write to cached pages that have been backed up with read-only files and thus can increase their privileges on the system. This is due to the use of a partially uninitialized memory of the pipe buffer structure in its construction. This means that the new member is insufficiently zero-initialized, which causes flag values to be out of date. This is exploited by an attacker to gain write access to cached pages, which are, however, originally marked read-only. This vulnerability gives an attacker several ways to gain root privileges, such as through unauthorized creation of new cron jobs, SUID binary hijacking, or by modifying the /etc/passwd file. [48]

An exploit has been found for replay cases and can be used in the following steps. First, the ELF executable needs to be downloaded on the system. Then you need to change the permissions for the file using the chmod file and see if your system is compromised by executing the file.

Listing 2.22: Setting permission of file and executing them.

```
$ chmod 777 traitor-amd64
$ ./traitor-amd64
```

If a vulnerable version of the Linux Kernel is found on the system the exploit can be run as follows:

Listing 2.23: Running exploit.

```
$ ./traitor-amd64 --exploit kernel:CVE-2022-0847
```

If the exploit is successful after entering the following commands the user should get root privileges and have a root ID:

Listing 2.24: After exploit.

```
# whoami
root
# id
uid=0(root) gid=0(root) groups=0(root)
```

This exploit gives the user root privileges on the system. The vulnerability will not be discussed in more detail as vulnerabilities are not the focus of this paper. The following paragraphs will describe how various SELinux settings can or cannot mitigate this privilege escalation.

Inactive SELinux

In this configuration, SELinux is inactive and thus there are no mechanisms that can mitigate this privilege escalation. An attacker gains full privileges to the system and can compromise the machine. A small change would occur in the permissive where SELinux is enabled but policies are not required so all unauthorized accesses are logged.

Unconfined_u

This security context is used on most systems running SELinux by default. The user has the security context unconfined_t. The problem is that root has a designated security context of unconfined_t i.e. the same as a regular user and for this context a large part of the rules in the policies are enabled.

So if an ordinary user who is not in the sudo group runs an exploit, they gain root privileges and can compromise the system as well.

Basic_u

This is already a SELinux user that is created by us and has templates created by us in its policies. He should have the least authority of the three users that were created for this job, and thus the SELinux mechanism should mitigate any privilege escalation the most.

Again, after the exploit is triggered, the user becomes root and has a root ID. If we look at the output of the following command we see this:

Listing 2.25: ID of basic_user.

```
# id
uid=0(root) gid=0(root) groups=0(root)
context=basic_u:basic_r:basic_t:s0
```

You can see the extra security context of a restricted basic user. Due to the fact that the SELinux server is running over classic discrete access control, the attacker has no chance to exploit root privileges on this account because root is still running in the security context of this user. The SELinux system would have to be inactive or in permissive mode for an attacker to gain root privileges. Thus, the policies we developed work against privilege escalation and can mitigate it on the SELinux side. So it always depends to what extent a given user is restricted based on that extent an attacker gains privileges. For a basic user, there is no way to use the sudo utility.

Advanced_u

The same situation occurs with another user who is more advanced and can already use the sudo utility, but even in this case the attacker does not gain more power than an ordinary user thanks to the SELinux mechanism. The only advantage the attacker gains is that the functionality that the user runs via sudo can be run as root without authenticating the user.

Admin_u

For an admin, the situation will be similar, except that it has a much larger range of functionality executable via sudo that could be exploited by an attacker.

2.4.2 Pwnkit

Another priviledge escalation that has been reproduced on our system is a vulnerability codenamed "PwnKit". This vulnerability can be found on a wide range of Linux distributions and is also found on the Red Hat Enterprise Linux distribution. As with the "DirtyPipe" vulnerability, an attacker can obtain root privileges.

Unlike the previous priviledge escalation, here the Kernel kernel is not affected, but a Polkit system component specifically pkexec. This vulnerability has been found to have existed for 12 years and has been in every version of Polkit released in that time.

Polkit is a component for managing system-wide permissions in Unix-like operating systems. It provides an organized way for non-privileged processes to communicate with privileged processes. Polkit provides a level of centralized system policy control.

A component of polkit is pkexec, which allows a privileged user to run commands as another user and is an alternative to the sudo tool.[49]

How PwnKit works

This vulnerability allows a user to execute the pkexec executable and pass it a specific set of environment variables that can execute an arbitrary library file. This allows an unprivileged attacker to invoke PolicyKit and force it to execute an attacker-controlled .so file on the file system. After this exploit, the attacker is able to perform privilege escalation from an ordinary user to a user with root privileges.[50]

An exploit has been found for this priviledge escalation, which you can download using this link. The source files that need to be built can be downloaded and then the exploit can be run. All the steps that have been written here are here:

Listing 2.26: Steps to reproduce PwnKit exploit.

```
[basic_user@thesis-rhel9 PwnKit-Exploit]$ make
cc -Wall    exploit.c   -o exploit
[basic_user@thesis-rhel9 PwnKit-Exploit]$ whoami
basic_user
[basic_user@thesis-rhel9 PwnKit-Exploit]$ ./exploit
Current User before execute exploit
hacker@victim$whoami: basic_user
[+] Enjoy your root if exploit was completed succesfully
bash: /root/.bashrc: Permission denied
bash-5.1# id -Z
basic_u:basic_r:basic_t:s0
bash-5.1# whoami
root
bash-5.1#
```

However, this exploit is valid for Polkit polkit-0.117-7.el9.x86_64 and earlier. A temporary workaround for this vulnerability was to issue this command, which removes the pkexec setuid bit.

Listing 2.27: Temporary workaround for PwnKit.

```
$ chmod 0755 /usr/bin/pkexec
```

Currently, current versions of polkit no longer contain this vulnerability.

And how can SELinux mitigate this vulnerability?

The impact of this priviledge escalation is exactly the same as the first vulnerability discussed. SELinux can therefore mitigate effectively depending on which

restricted user is mapped to the Linux user on the system. This means that the user basic_u who only provides basic functionality can mitigate the most and the user admin_u who has the most authority can mitigate the least, but this does not take into account systems that do not use our restricted users. In these cases, SELinux cannot effectively mitigate privilege escalation.

In conclusion, SELinux in our configuration with newly created users can therefore successfully mitigate an attacker's attempts to elevate privileges on the system. We have demonstrated this with the kernel vulnerability example and also with the system component vulnerability example. SELinux with our limited users is thus an interesting option to secure possible privilege escalation on the system. Still on the issue of vulnerabilities that we have not been able to reproduce, there is an assumption that SELinux would be able to mitigate all these vulnerabilities because the security server always runs on top of the classic discrete access control that is on the system.

2.5 Future improvements

Although we were able to create functional security policies on the Red Hat Enterprise Linux operating system for everyday use, this work was not able to cover all the functionality that this system offers to users. Full coverage of the functionality was beyond the scope of this work. And as a possible future enhancement to the restricted user policies, the policies can be extended to cover additional software on the system.

Another possible enhancement is to make role switching with the newrole tool available for advanced_u and admin_u users, where a user who has the security context of advanced_u will be able to gain more privileges and switch to the admin_r role. This will give them all the privileges on the system that admin_u has but with the associated security risk.

Another possible policy enhancement is to create additional confined users that would more specify the use of the system, such as a confined user that specializes in working with logs or a confined user that specializes in working with SELinux technology.

3 Conclusion

In this thesis, we addressed how to handle user rights using SELinux technology in Red Hat Enterprise Linux. First, the general security mechanisms and access control found on GNU/Linux systems were discussed. Also, the classic discrete access control found on all GNU/Linux operating systems was explained and then the principle of Mandatory Access Control, which can be added as a module to the GNU/Linux kernel, was discussed.

Furthermore, the SELinux technology, which is crucial for our work, was discussed in more detail. Regarding SELinux technology we described more about its basics, how it enforces security on the system. What the different elements of SELinux technology exist and also how it restricts services from working on the system and how we can directly restrict GNU/Linux users on the system. Describing the ways of restricting users on the system was important because in the practical part, restricting users is the goal of the whole thesis.

Therefore, the overall objective of the work was to design security rules based on typical user activities and further implement these rules and then test the functionality of the security policies. First, we did some research on typical uses of the GNU/Linux OS. The survey resulted in a list of typical uses of the OS, which was divided into several logical units such as basic operations or security on the system.

The survey was followed by a description of our proposed architecture of how the new confined users will be structured and what structure will be used in the security policies to allow the most variable cutomization. It was decided that the security rules that will then be added to the SELinux user policies will be encapsulated in templates that can thus be freely used. It was decided to create three confined SELinux users, which are different in functionality and are intended for different typical GNU/Linux users.

In the implementation part, we aim to implement the principles and mechanisms of the new policies proposed by us. We first describe how to create a custom policy for a restricted user. The steps from the creation of a GNU/Linux user, a SELinux user, to the subsequent mapping of the two mechanisms and the linking of the GNU/Linux user to the SELinux security context are broken down.

This tutorial was followed by a description of the functionalities of each template that was created. It described what resource and data accesses we need to add to the template to enable the user. Several of these templates were created and described and we won't list them all in the conclusion, but the more important templates that were created are the user login template and then the GUI template.

Thanks to the implementation of the new template, we have discovered several errors in the SELinux security policies that need to be fixed and can already benefit

all users using the technology. For example, we discovered a bad label for temporary systemd files that were tagged as user_tmp_t, causing problems in policies.

After that, it was still necessary to create individual SELinux users and apply the templates created to them. Once the users were successfully implemented and configured on the system, the intended functionality of the individual confined users was tested to see if the individual users had more powers than declared. In the penultimate part of the practical work, we tried using our new SELinux users to mitigate the latest vulnerabilities that have emerged. And in the last part we outlined possible improvements to our confined users for the future.

SELinux technology has proven that it can be an interesting option to limit users and mitigate possible attacks, which recently also aim at controlling the system through user compromise. Thanks to our confined users, this danger can be minimized, making it harder for attackers to penetrate the system. And with further improvements that can be made in the future, this technology can even better secure operating systems that are used around the world by many users.

Bibliography

[1] *Red Hat Product Security risk report: 2020* [online]. Raleigh: Red Hat, 2021 [cit. 2021-11-03]. Available from: https://www.redhat.com/en/resources/product-security-risk-report-2020

[2] *Chapter 2. Introduction* [online]. Raleigh: Red Hat, 2020 [cit. 2021-11-03]. Available from: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/chap-security-enhanced_linux-introduction

[3] *Chapter 2. Introduction* [online]. Raleigh: Red Hat, 2020 [cit. 2021-11-03]. Available from: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/chap-security-enhanced_linux-introduction

[4] *Red Hat Product Security risk report: 2020* [online]. Raleigh: Red Hat, 2021 [cit. 2021-11-03]. Available from: https://www.redhat.com/en/resources/product-security-risk-report-2020

[5] *Chapter 2. Introduction* [online]. Raleigh: Red Hat, 2020 [cit. 2021-11-03]. Available from: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/chap-security-enhanced_linux-introduction

[6] *Chapter 2. Introduction* [online]. Raleigh: Red Hat, 2020 [cit. 2021-11-03]. Available from: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/chap-security-enhanced_linux-introduction

[7] *How Secure Is Linux?* [online]. Upper Saddle River, N.J.: Guardian Digital, 2021 [cit. 2021-11-04]. Available from: https://linuxsecurity.com/features/how-secure-is-linux

[8] KŇAŽEKOVÁ, Nikola. *SECURITY OF RED HAT ENTERPRISE LINUX BASED OPERATING SYSTEMS*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Prof. Ing. Dan Komosný, Ph.D.

[9] KŇAŽEKOVÁ, Nikola. *SECURITY OF RED HAT ENTERPRISE LINUX BASED OPERATING SYSTEMS*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Prof. Ing. Dan Komosný, Ph.D.

[10] *Overview of Linux Kernel Security Features* [online]. USA: The Linux Foundation, 2013 [cit. 2021-11-05]. Available from: https://www.linux.com/training-tutorials/overview-linux-kernel-security-features/

[11] *Special File Permissions (setuid, setgid and Sticky Bit)* [online]. USA: Oracle Corporation, 2010 [cit. 2021-11-05]. Available from: https://docs.oracle.com/cd/E19683-01/816-4883/secfile-69/index.html

[12] *The SCO Group, Inc. UnixWare 7 Documentation. Discretionary access control DAC: permission bits* [online]. [cit. 2021-11-06]. Available at: http://uw714doc.xinuos.com/en/SEC_file/_Discretionary_Access_Control_DAC_perms.html

[13] *Sticky bit is only for directories?* [online]. USA: StackOverflow, 2017 [cit. 2021-12-05]. Dostupné z: https://stackoverflow.com/questions/41435657/sticky-bit-is-only-for-directories-i-found-it-could-be-on-files-weird

[14] *Securing Linux with Mandatory Access Controls* [online]. USA: The Linux Foundation, 2005 [cit. 2021-11-05]. Available from: https://www.linux.com/news/securing-linux-mandatory-access-controls/

[15] *Securing Linux with Mandatory Access Controls* [online]. USA: The Linux Foundation, 2005 [cit. 2021-11-05]. Available from: https://www.linux.com/news/securing-linux-mandatory-access-controls/

[16] *Linux Security Module Usage* [online]. USA: The Linux Kernel, 2021 [cit. 2021-11-22]. Available from: https://www.kernel.org/doc/html/v4.16/admin-guide/LSM/index.html

[17] *Securing Linux with Mandatory Access Control* [online]. GOTHENBURG, SWEDEN: Scionova, 2019 [cit. 2021-11-06]. Available from: https://www.scionova.com/2019/04/08/securing-linux-with-mandatory-access-control/

[18] *Linux Security Module Usage* [online]. USA: The Linux Kernel, 2021 [cit. 2021-11-22]. Available from: https://www.kernel.org/doc/html/v4.16/admin-guide/LSM/index.html

[19] HAINES, Richard. The SELinux Notebook: (4th Edition) [online]. 2014 [cit. 2021-11-06]. Available at: http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

[20] HAINES, Richard. The SELinux Notebook: (4th Edition) [online]. 2014 [cit. 2021-11-06]. Available at: http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

[21] *Security Context* [online]. USA: Red Hat, 2014 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/NB_SC

[22] *Security Context* [online]. USA: Red Hat, 2014 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/NB_SC

[23] *Security Context* [online]. USA: Red Hat, 2014 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/NB_SC

[24] *Security Context* [online]. USA: Red Hat, 2014 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/NB_SC

[25] HAINES, Richard. The SELinux Notebook: (4th Edition) [online]. 2014 [cit. 2021-11-06]. Available at: http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

[26] *AVCRules* [online]. USA: Red Hat, 2015 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/AVCRules

[27] *AVCRules* [online]. USA: Red Hat, 2015 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/AVCRules

[28] *AVCRules* [online]. USA: Red Hat, 2015 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/AVCRules

[29] *AVCRules* [online]. USA: Red Hat, 2015 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/AVCRules

[30] *AVCRules* [online]. USA: Red Hat, 2015 [cit. 2021-11-09]. Available from: https://selinuxproject.org/page/AVCRules

[31] VERMEULEN, S. *SELinux Cookbook*. Packt Publishing, 2014, 214 s. ISBN 978-1783989669.

[32] HAINES, Richard. The SELinux Notebook: (4th Edition) [online]. 2014 [cit. 2021-11-06]. Available at: http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

[33] HAINES, Richard. The SELinux Notebook: (4th Edition) [online]. 2014 [cit. 2021-11-06]. Available at: http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

[34] HAINES, Richard. The SELinux Notebook: (4th Edition) [online]. 2014 [cit. 2021-11-06]. Available at: http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

[35] *Using SELinux* [online]. Raleigh: Red Hat, 2021 [cit. 2021-11-10]. Available from: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9-beta/html-single/using_selinux/index#managing-confined-and-unconfined-users_using-selinux

[36] *Using SELinux* [online]. Raleigh: Red Hat, 2021 [cit. 2021-11-10]. Available from: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9-beta/html-single/using_selinux/index#managing-confined-and-unconfined-users_using-selinux

[37] KŇAŽEKOVÁ, Nikola. *SECURITY OF RED HAT ENTERPRISE LINUX BASED OPERATING SYSTEMS*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Prof. Ing. Dan Komosný, Ph.D.

[38] *Linux Documentation Project Guides* [online]. USA: Linux Documentation Project, 2014 [cit. 2021-11-22]. Available from: https://tldp.org/guides.html

[39] *Welcome to Freedom* [online]. USA: Red Hat, 2021 [cit. 2021-11-15]. Available from: https://getfedora.org/

[40] *Ways of logging to Linux machine* [online]. USA: Stack&Exchange, 2014 [cit. 2021-11-16]. Available from: https://unix.stackexchange.com/questions/109999/ways-of-logging-to-linux-machine

[41] *34 Linux Basic Commands Every User Should Know* [online]. –: Hostinger, 2021 [cit. 2021-11-16]. Available from: https://www.hostinger.com/tutorials/linux-commands

[42] *The most obvious user for Linux isn't who you think* [online]. USA: Technology Advice, 2021 [cit. 2021-11-16]. Available from: https://www.techrepublic.com/article/the-most-obvious-user-for-linux-isnt-who-you-think/

[43] *Linux System Admin Command* [online]. USA: JavaTpoint, 2021 [cit. 2021-11-16]. Available from: https://www.javatpoint.com/linux-system-admin-commands

[44] *5 everyday sysadmin tasks to automate with Ansible* [online]. USA: Red Hat, 2021 [cit. 2021-11-16]. Available from: https://opensource.com/article/21/3/ansible-sysadmin

[45] *Red Hat Enterprise Linux (RHEL) 9 Beta se zaměřuje na automatizaci* [online]. Praha: Root.cz, 2021 [cit. 2021-11-11]. Available from: https://www.root.cz/zpravicky/red-hat-enterprise-linux-rhel-9-beta-se-zameruje-na-automatizaci/

[46] *Systemd System and Service Manager* [online]. USA: Freedesktop, 2021 [cit. 2021-11-22]. Available from: https://www.freedesktop.org/wiki/Software/systemd/

[47] *New Linux bug gives root on all major distros, exploit released* [online]. USA: Bleeping Computer, 2022 [cit. 2022-04-21]. Dostupné z: https://www.bleepingcomputer.com/news/security/new-linux-bug-gives-root-on-all-major-distros-exploit-released/

[48] *Linux system service bug gives root on all major distros, exploit released* [online]. USA: BleepingComputer.coM, 2022 [cit. 2022-04-26]. Dostupné z: https://www.bleepingcomputer.com/news/security/linux-system-service-bug-gives-root-on-all-major-distros-exploit-released/

[49] *Linux system service bug gives root on all major distros, exploit released* [online]. USA: BleepingComputer.coM, 2022 [cit. 2022-04-26]. Dostupné z: https://www.bleepingcomputer.com/news/security/linux-system-service-bug-gives-root-on-all-major-distros-exploit-released/

[50] *Linux system service bug gives root on all major distros, exploit released* [online]. USA: BleepingComputer.coM, 2022 [cit. 2022-04-26]. Dostupné z: https://www.bleepingcomputer.com/news/security/linux-system-service-bug-gives-root-on-all-major-distros-exploit-released/

Symbols and abbreviations

ALSA	Advanced Linux Sound Architecture
CVE	Common Vulnerabilities and Exposures
DAC	Discretionary Access Control
ELF	Executable and Linkable Format
FLASK	Flux Advanced Security Kernel
GUI	Graphical User Interface
LSM	Linux Security Modules
MAC	Mandatory Access Control
MCS	Multi-Category Security
MLS	Multi-Level Security
RBAC	Role-Based Access Control
OS	operating system
RHEL	Red Hat Enterprise Linux
SaaS	Software as a Service
SELinux	Security-Enhanced Linux
TE	Type enforcement
USB	Universal Serial Bus