BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

MASTER'S THESIS

Brno, 2020 Nikola Kňažeková

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

SECURITY OF RED HAT ENTERPRISE LINUX BASED OPERATING SYSTEMS
BEZPEČNOST OPERAČNÍCH SYSTÉMŮ ZALOŽENÝCH NA DISTRIBUCI RED HAT ENTERPRISE LINUX

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Nikola Kňažeková**
AUTOR PRÁCE

SUPERVISOR — **prof. Ing. Dan Komosný, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2020

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Master's Thesis

Master's study field **Information Security**

Department of Telecommunications

Student: Nikola Kňažeková ***ID:*** 208304

Year of study: 2 ***Academic year:*** 2019/20

TITLE OF THESIS:

Security of Red Hat Enterprise Linux based operating systems

INSTRUCTION:

Analyse the vulnerabilities of operating systems based on the Red Hat Enterprise Linux distribution. Cover vulnerabilities from previous 5 years. Evaluate whether the studied attacks could have been blocked by the SELinux technology. Propose security measures and implement them using the Ansible automation tool. Verify the proposed defence against the selected attacks.

Within the semestral project, implement everything up to the proposal of security measures, including automation of their deployment. Within the diploma thesis, verify the reliability of defence. In case of any problems, modify the security measures retrospectively.

RECOMMENDED LITERATURE:

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

[2] REMPEL, C. A system administrator's guide to getting started with Ansible - FAST! Red Hat blog [online]. 12. 3. 2018 [cit. 2019-09-05]. Dostupné z: https://tinyurl.com/yyasqznr

Date of project specification: 3.2.2020 ***Deadline for submission:*** 1.6.2020

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

prof. Ing. Jiří Mišurec, CSc.
Subject Council chairman

ABSTRACT

This master's thesis focuses on increasing the security of Red Hat Enterprise Linux derived operating systems, based on the vulnerabilities analyzed over the last 5 years. The theoretical part describes the weaknesses and vulnerabilities, and the basic security mechanisms in Linux, with a focus on a SELinux technology. SELinux technology is shipped with the operating systems Red Hat Enterprise Linux, Fedora, and CentOS. In the practical part, based on the analyzed vulnerabilities, the configuration of the SELinux technology was designed. The design describes the elements that will be configured, namely SELinux booleans, SELinux modules, and SELinux users, focused on memory protection, escalation of privileges, code execution, data leakage, and restriction of processes and users. Based on the suggestions, a configuration was created in the Ansible configuration tool, which aims to allow the user to simply and quickly configure systems. In addition, two more configurations have been created that will allow users to revert the SELinux configuration to a previous state or lockdown the SELinux configuration. Subsequently, the impact of the configurations on the usability of the system was verified, and the found bugs were fixed or reported. The last part verifies the functionality of the configuration against exploit vulnerabilities.

KEYWORDS

Discretionary Access Control, Mandatory Access Control, Red Hat Enterprise Linux, Security, SELinux Technology, Vulnerability

ABSTRAKT

Táto diplomová práca sa zameriava zvyšovanie bezpečnosti v operačných systémoch založených na Red Hat Enterprise Linux, na základe analyzovaných zraniteľnosti za posledných 5 rokov. V teoretickej časti sú popísané slabiny a zraniteľnosti, základné bezpečnostné mechanizmy v Linuxe, so zameraním na technológiu SELinux. Technológia SELinux je súčasťou operačných systémov Red Hat Enterprise Linux, Fedora a CentOS. Na základe analyzovaných zraniteľností bola v praktickej časti navrhnutá konfigurácia technológie SELinux. V návrhu sú popísané prvky, ktoré sa budú konfigurovať a tými sú SELinuxové booleany, SELinuxové moduly a SELinuxoví užívatelia, so zameraním na ochranu pamäte, eskalovanie privilégií, spúšťaniu kódu, úniku dat a obmedzenie procesov a užívateľov. Na základe návrhov bola vytvorená konfigurácia v konfiguračnom nástroji Ansible, ktorej cieľom je umožniť užívateľovi jednoducho a rýchlo nakonfigurovať hosťa. Okrem nej boli vytvorene ďalšie dve konfigurácie, ktoré umožnia vrátiť systém do predchádzajúceho stavu alebo uzamknúť SELinuxovú konfiguráciu. Následne sa overoval dopad konfigurácií na použiteľnosť systému a nájdené chyby boli opravené alebo nahlásené. Posledná časť overuje funkčnosť konfigurácie pred zneužitím zraniteľností.

KLÍČOVÁ SLOVA

Bezpečnosť, Povinné riadenie prístupu, Red Hat Enterprise Linux, SELinux Technológia, Voliteľné riadenie prístupu, Zraniteľnosť

KŇAŽEKOVÁ, Nikola. *Security of Red Hat Enterprise Linux based operating systems*. Brno, Rok, 98 p. Master's Thesis. Brno University of Technology, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Advised by doc. Ing. Dan Komosný, Ph.D.

Typeset by the `thesis` package, version 3.05; `http://latex.feec.vutbr.cz`

ROZŠÍRENÝ ABSTRAKT

Zero-day útoky sú fenoménom dnešnej doby. Informačný systém môže byť napadnutý útočníkom dávno predtým, než sa verejnosť vôbec dozvie že nejaká zraniteľnosť existuje a aplikujú sa bezpečnostné záplaty. Z tohto dôvodu je potreba používať bezpečnostné technológie, ktoré môžu zablokovať alebo aspoň obmedziť dopady zero-day útokov, ako je napríklad technológia SELinux založená na koncepte povinného riadenia prístupu.

V tejto diplomovej práci sa zameriavam na bezpečnostnú technológiu SELinux, dodávanú s operačnými systémami založenými na Red Hat Enterprise Linux. Cieľom práce je analyzovanie zraniteľností na tieto operačné systémy za posledných 5 rokov. Navrhnutie a implementácia striktnejšej konfigurácie technológie SELinux v automatizačnom nástroji Ansible. Účelom stritknejšej konfigurácie je zabránenie vybraným útokom alebo obmedzenie ich dopadov a následné otestovanie jej účinnosti voči vybraným útokom.

Teoretická časť tejto diplomovej práce oboznamuje čitateľa so zraniteľnosťami, slabinami a databázami kde ich nájsť. Ďalej sú popísané bezpečnostné mechanizmy v linuxových operačných systémoch. Pozornosť je venovaná technológií SELinux, v rámci ktorej sú popísané aj základné nástroje pre jej konfiguráciu. Technológia SELinux jej dodávaná v operačných systémoch založených na Red Hat Enterprise Linux, ktorými sú Fedora, CentOS a Red Hat Enterprise Linux.

V praktickej časti boli zistené najčastejšie zraniteľnosti a ich technické dopady z databázy poskytnutej Red Hat Product Security, ktorá obsahovala vyše 8 000 údajov za posledných 5 rokov. Na základe technických dopadov bola navrhnutá konfigurácia technológie SELinux, ktorá sa zameriava na ochranu pamäte, ďalej zabráneniu eskalácie privilégií, spúšťaniu kódu, úniku dát a používaniu unconfined a permissive domén, v ktorých mali procesy a užívatelia takmer neobmedzený prístup v systéme. Taktiež boli navrhnuté ďalšie dve konfigurácie. Prvá, ktorá umožňuje vrátiť SELinux konfiguráciu do predošlého stavu a tak v prípade potreby anulovať všetky zmeny. Naopak druhá konfigurácia zas uzamkne SELinux konfiguráciu a znemožní akékoľvek modifikácie či vypnutie technológie SELinux.

Všetky konfigurácie boli vytvorené v automatizačnom nástroji Ansible, ktorý umožňuje hromadnú konfiguráciu hostov. SELinuxová technológia bola nastavovaná pomocou modifikovania stavu SELinuxových boolean, modulov a užívateľov. V hlavnej konfigurácií sa na začiatku exportujú lokálne nastavenia SELinuxu a uložia sa do home adresára konfigurovaného hosťa.

Následne sa pomocou nastavenia SELinuxových boolean zakazuje zdieľanie súborov, pripojenie rôznych služieb k portom, spúšťanie častí pamäte a zásobníku. Ďalej sú mapovaní linuxoví užívatelia do SELinuxových užívateľov. Taktiež sú deaktivované unconfined a permissive domény, ktoré umožňovali neobmedzený prístup pre

užívateľov a procesy, na ktoré nebola aplikovaná SELinuxová politika. Konfigurácia je vytvorená univerzálne a vyžaduje miernu interakciu užívateľom, podľa jeho potrieb, popísané v kapitolách 5., 7. a 8. Napríklad je potrebné určiť, ktorí užívateľa v systéme budú mapovaní do SELinuxových užívateľov. Alebo je potrebné nastavenie boolean pre služby, ktoré sa na konfigurovanom hosťovi budú používať. Pri navrhovaní konfigurácie bola objavená chyba v SELinuxovej politike pre Flatpak službu, opravená a bol vytvorený tzv. "pull request" do Flatpak repozitára na GitHube.

Druhá, Revert konfigurácia umožňuje vrátenie SELinuxovej konfigurácie do predchádzajúceho stavu. Na začiatku sa na konfigurovanom hosťovi aplikujú základné nastavenia SELinuxovej technológie. Následne sa importujú lokálne nastavenia, ktoré boli uložené počas spúšťania hlavnej konfigurácie.

Posledná konfigurácia, ktorá má za cieľ uzamknutie SELinuxových nastavní, používa booleany z rady secure_mode. Tieto booleany užívateľom mapovaným do SELinuxových užívateľov zakazujú získať oprávnenia superužívateľa, nahrať vlastné SELinuxové moduly a akokoľvek upravovať nastavenia SELinuxu. Po aplikovaní tejto konfigurácie nebude možné vrátiť zmeny, dokým sa SELinux nevypne alebo nenastaví do permissive módu pridaním kernelových parametrov počas bootovania, tak ako je popísané v 8. kapitole.

Po vykonaní hlavnej konfigurácie sa používateľ môže rozhodnúť, či chce vrátiť SELinxové nastavenia do predchádzajúceho stavu pomocou Revert konfigurácie alebo či si chce uzamknúť SELinuxové nastavenia pomocou konfigurácie na zablokovanie systému. Konfigurácie sú dostupné na gitlabe:
https://gitlab.com/5umm3r15/selinux-hardening.git

V ďalšej časti diplomovej práce sú sledované dopady konfigurácie na použiteľnosť systému a aplikácií a doporučenia pre užívateľov. Prvým významným dopadom je, že užívatelia nebudú môcť používať grafické rozhranie, pokiaľ bude zapnutá booleana deny_execmem. V tejto časti je aj popis ako ju vypnúť. Následne sú charakterizovaní SELinuxoví užívateľia ktorí môžu získať oprávanenia superužívateľa a potrebné kroky v prípade SELinuxového užívateľa staff_u k ich získaniu. Okrem toho je tú popísaná konfigurácia na uzamknutie SELinuxového systému. Jeden problém charakterizuje SELinuxovú booleanu secure_mode, ktorý by mal zabrániť tomu, aby SELinuxový užívateľ staff_u získal oprávnenia superužívateľa. Počas testovania povolenej secure_mode booleany a SELinuxového užívateľa staff_u sa našla možnosť ako túto booleanu obísť. Môže sa jednať o bezpečnostnú chybu v SELinuxovej politike, preto bola nahlásená na Red Hat Bugzilla. Ďalej je uvedený spôsob, ako manuálne zamedziť SELinuxovému užívateľovi staff_u k získaniu superužívateľských oprávnení. Následne je popísané, ako zmeniť stav SELinuxu pridaním kernelových parametrov, ak je povolená booleana secure_mode_policyload. Táto

booleana je zodpovedná za uzamknutie SELinuxovej konfigurácie, ktorú nemožno upravovať z užívateľského rozhrania. Posledná časť tejto kapitoly porovnáva výkon systému pri zapnutom SELinuxe a vypnutom SELinuxe. Prvým opisovaným nástrojom je systemd-analyze, ktorý sleduje rýchlosť zapínania systému. So zapnutým SELinuxom sa rýchlosť zapínania systému znížila o 14 %. Ďalším nástrojom je UnixBench obsahujúci rôzne testy, ktoré merajú čas kopírovania súborov, tvorby procesov, spúšťanie skriptov, systémových volaní, zapisovanie a čítanie z rúr. Nástroj UnixBench počíta indexy z výsledkov testov, kde vyšší index znamená lepší výkon. Indexy sa so zapnutým SELinuxom znížili o 12 % až 24 %, kde najmenší rozdiel bol v testoch systémových volaní a najväčší rozdiel nastal pri testoch zápisu a čítania z rúr.

V poslednej kapitole je popísaná funkčnosť nasadenej konfigurácie proti zneužitiu zraniteľností. Taktiež sú tu popísané prvky SELinuxovej technológie, ktoré zabránili zneužitiu zraniteľností alebo zmiernili ich dopady. Najväčším problémom bolo nájsť zraniteľné verzie systémov a aplikácií alebo funkčný exploit. Prvý popísaný prípad zobrazuje eskalovanie privilégií zneužitím zraniteľnosti v ptrace traceme. Zraniteľnosť bola úspešne zablokovaná mapovaním linuxových užívateľov do SELinuxových užívateľov. Ďalším spôsobom ako zabrániť zneužitiu tejto zraniteľnosti je zapnutie booleany deny_ptrace. Druhá popísaná zraniteľnosť, ktorú je možné blokovať zapnutím booleany deny_ptrace, je tzv. Dirty COW, ktorá zneužíva chybu v copy-on-write a taktiež umožňuje eskalovanie privilegií. Nasledovne bol popísaný Path traversal útok na Apache web server, ktorým je možné pristúpiť k súborom a adresárom mimo koreňového adresára webu. Základné nastavenie SELinuxu zmiernilo dopad tohto útoku a to tým, že útočník môže pristúpiť len k súborom, ku ktorým má povolenie pristupovať Apache proces v SELinuxovej politike. Nemôže zobraziť napríklad súbory v home adresári, ku ktorým to Apache proces nemá v politike definované. Základná SELinuxová konfigurácia zabránila aj zraniteľnosti vo funkcii mmap ktorá umožňovala dereferencovanie null pointera. Všetky opísané nastavenia SELinuxu, ktoré chránia pred zneužitím zraniteľností, sú súčasťou nasadenej konfigurácie SELinuxu.

Posledná kapitola dokázala význam SELinuxu v prípade zero-day útokom. SELinux zabránil niektorým útokom v základnej konfigurácií, na iné bolo treba uplatniť striktnejšie nastavenia, ktoré sú zahrnuté v navrhnutej konfigurácií. Užívatelia si môžu jednoducho nakonfigurovať striktnejšie nastavenia SELinuxu na hostoch pomocou Ansible nástroja.

DECLARATION

I declare that I have written the Master's Thesis titled "Security of Red Hat Enterprise Linux based operating systems" independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the thesis and listed in the comprehensive bibliography at the end of the thesis.

As the author I furthermore declare that, with respect to the creation of this Master's Thesis, 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., Section 2, Head VI, Part 4.

Brno ....

author's signature

ACKNOWLEDGEMENT

I am extremely grateful to my supervisor prof. Ing. Dan Komosný Ph.D. for his guidance and professional support during the work on this thesis. Special thanks to colleagues from Red Hat, Bc. Lukas Vrabec, Senior Software Engineer, for his excellent guidance and collaboration in sharing his broad knowledge and experience in SELinux Technology and Alexander Jacocks, Staff Solutions Architect, for all the support, help, and advice. Moreover, I want to express my sincerest thanks to all my colleagues from the Security Controls team. Finally, and most importantly, I would like to thank my partner and family for all of the support they have given me.

Contents

List of Figures

List of Tables

Listings

Introduction

The number of vulnerabilities is growing every day. Some vulnerabilities are not dangerous, while others have a critical impacts that can lead to escalating privileges or code execution. In the case of critical vulnerabilities, it is necessary to react immediately and patch the system. The problem are zero-day attacks, when patches are not yet available, or the developers do not even know about the vulnerability, but the attackers are already exploiting it. Due to such cases, it is advisable to use security technology based on the Mandatory Access Control concept, like SELinux. SELinux is used in various operating systems to determine access defined by the SELinux policy, and what is not specifically allowed is denied. However, the default SELinux configuration may be too benevolent, to prevent users from troubleshooting issues, or in the worst cases, disabling SELinux. But more usability means less security.

This master's thesis is focused on designing a SELinux technology configuration to increase the security of Red Hat Enterprise Linux based operating systems, according to analyzed attacks over the past five years. The first chapter describes the weaknesses and vulnerabilities, and the standards by which they identify. Understanding the difference between vulnerabilities and weaknesses is essential for the next chapters. The second chapter describes security mechanisms in Linux, Discretionary Access Control, Mandatory Access Control, and SELinux technology. The main elements of SELinux technology are discussed as they will be handled during configuration. Subsequently, systems using SELinux technology are described. The third chapter shows the base description of Red Hat Enterprise Linux based operating systems: Fedora, Red Hat Enterprise Linux, and CentOS. Moreover, the differences between them are also decomposed. The SELinux technology configuration will be designed for these operating systems. The fourth chapter describes the most frequent weaknesses and technical impacts over the last 5 years. The fifth chapter designs the configuration of SELinux based on technical impacts, managed with SELinux booleans, SELinux modules, and SELinux user mapping. Also designed are reversion options, to go back to a previous state, and system lockdown options to prevent from managing SELinux state and configuration. The sixth chapter introduces the automation tool Ansible and in the seventh chapter the configuration of SELinux in Ansible is described, configuration for reversion options, and configuration of system lockdown options. The eighth chapter focuses on the security impacts on usability and performance. The ninth chapter presents features applied in configurations against exploited vulnerabilities and their impact with SELinux in permissive mode, SELinux with default configuration and SELinux with deployed stricter configuration.

1 Security weaknesses and vulnerabilities

It is necessary to explain vulnerabilities, weaknesses and the relationships between them, and the standards that are necessary for their clear identification. A weakness is any architecture, design, code, or implementation error that can lead to a vulnerability being exploited by an attacker [1]. A vulnerability is a weakness found in software or hardware that has a negative impact on confidentiality, integrity or availability, after exploitation [2].

1.1 Weaknesses

In 2005, MITRE created a Common weakness enumeration (CWE) list, containing common software weaknesses and classifying them according to their similarities and differences. Weaknesses, on the CWE list, have individual identifiers CWE-ID, where ID is a number. Common weakness enumeration entries are available at the MITRE website[1]. CWE list version 3.4.1 contains 808 weaknesses and 295 classes.

"Each weakness, if successfully exploited, can lead to one or more potential Technical Impacts:

- modify data,
- read data,
- DoS: unreliable execution,
- DoS: resource consumption,
- execute unauthorized code or commands,
- gain privileges / assume identity,
- bypass protection mechanism,
- hide activities [3].

1.2 Vulnerabilities

To identify known security vulnerabilities, in 1999 MITRE launched a Common Vulnerabilities and Exposures (CVE) dictionary. In the CVE standard, each vulnerability has an identification number, description, and reference. Each CVE identifier has form: CVE-year-number. Currently, the CVE list describes over 124 000 vulnerabilities available on CVE[2] website [4]. Each vulnerability has to have assigned particular weakness [1].

The time between vulnerability disclosure and vulnerability patching is called **window of exposure**. Zero-day attacks happen when exploits are released in the

[1] https://cwe.mitre.org
[2] https://cve.mitre.org

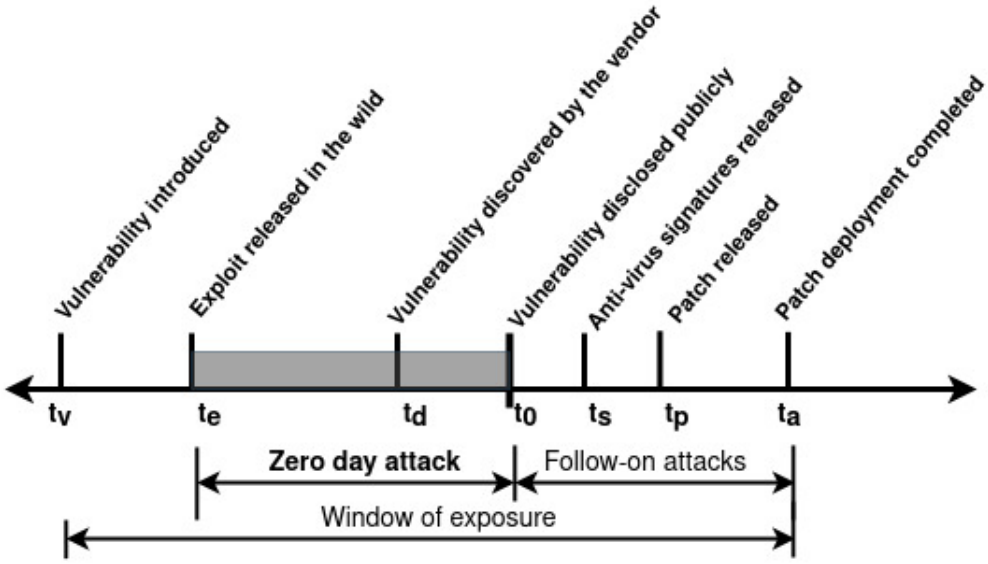

Fig. 1.1: Window of exposure. [5]

wild and not publicly known. When vulnerabilities are disclosed publicly, exploits of unpatched vulnerabilities are called follow-on attacks [5].

The CVE standard helps to find information about a particular vulnerability, for example in various databases. Vulnerability databases, unlike the CVE list, also contain risk, impact and other technical information [6].

National Vulnerability Database A free database, synchronized with the CVE dictionary, is provided by the National Vulnerability Database (NVD)[3]. NVD provides information and details about the vulnerability, such as technical impact, known affected software configurations, assigned weakness, references, etc. [7].

Red Hat CVE Database The Red Hat CVE database[4] contains CVE-IDs, impact and published dates. Each CVE-ID has a short description, statement from Red Hat, CWE-ID, score and references, included bug reference on Bugzilla. Red Hat Product Security provides security data, reports and metrics[5]. Many files are updated twice a day, such as a file with mapping CVE to CWE [8].

[3]https://nvd.nist.gov/vuln/
[4]https://access.redhat.com/security/security-updates/#/cve
[5]https://www.redhat.com/security/data/metrics/

2 Access Control Mechanisms in OS Linux

Access control mechanism is used to control which users or processes have access to system resources and data. Users and processes that initiate an access request are known as subjects. Files, devices and other resources to which access is requested are called objects [9]. In the OS Linux, the most common access control mechanisms are Discretionary Access Control and Mandatory Access Control [10].

2.1 Discretionary Access Control

Discretionary Access Control (DAC) is a standard Unix security model. Access control is based on file ownership. The owner of the file, a subject, has rights to manipulate with the file, an object. Each object has a permission bit that controls the access of a subject:

- owner,
- group,
- others [11].

Each object has a permission bit that represents operation:

- read - r,
- write - w,
- execute - x,
- no permissions are displayed with - [11].

Permission to the object are represented by three sets of three bits interpreted with characters or numbers:

Tab. 2.1: DAC users and permissions.

	user	group	others
-	r w x	r w x	r w x

Special permissions For executable files and directories can be used special permissions:

- **sticky bit** - **t** or **T**,
- **setuid** - **s** or **S**,
- **setgid** - **s** or **S** [12].

Sticky bit is a permission bit that doesn't allow a user to delete files or directories of other users. It is used for users not directly specified in the file permissions, and represented with the character **t** or **T** instead of **x**:

- the upper-case **T** represents a sticky bit, without execution bit,
- the lower-case **t** represents a sticky bit together with execution bit [13].

When **setuid** and **setgid** permissions are used, the user runs an executable file with privileges of the owner or group. Setuid and setgid permissions are represented with character **s** or **S** instead of **x** for user or group:

- the upper-case **S** represents setuid or setgid bit, without execution bit,
- the lower-case **s** represents setuid or setgid bit together with execution bit [13].

Setuid and **setgid** permissions can be security risks because *"a user can gain superuser privileges by executing a program that sets the user ID (UID) to root [12]."*

Tab. 2.2: Special bits.

r w x	r w x	r w x
SUID ↓	SGID ↓	Sticky ↓
r w s	r w s	r w t
user	group	others

Superuser - root, has the ability to manipulate any objects in the system. In the case of compromise of a privileged user or process, an attacker can control the entire system. To protect the entire system from compromise, the solution is mandatory access control. This is where access is defined in policy, and what is not allowed is prohibited [10].

2.2 Mandatory Access Control

Mandatory Access Control (MAC) is a security concept where access control decisions are based on security policy. Security Policy identifies the rules, which permit subjects to perform certain operations on objects [10].

At the beginning of the 21st century, several projects were set up to develop mandatory access control. For a simple implementation of MAC-based technologies, the Linux Security Modules (LSM) framework was developed [14]. Linux Security Modules framework was added in Linux kernel version 2.6. This framework allows implementation of various modules based on Access Control. LSM provides a series of hooks to allow modules to access kernel objects [15].

To control operations on kernel object, a function is called from the implemented LSM module, to perform the access control check. Based on the security rules defined in the module, access is allowed or denied [17].

2.3 Security-Enhanced Linux

Security-Enhanced Linux (SELinux) is a mandatory access control technology that uses the LSM framework. The first release of SELinux was in 2000, by the National Security Agency (NSA) [18].

The picture below shows permissions checks in a System Call, mechanism that provides the interface between an application and the Linux kernel [16].

Firstly DAC permissions are checked. If the system call is allowed, then LSM hooks are triggered to enable access control of security modules, such as SELinux. SELinux then determines access [18].

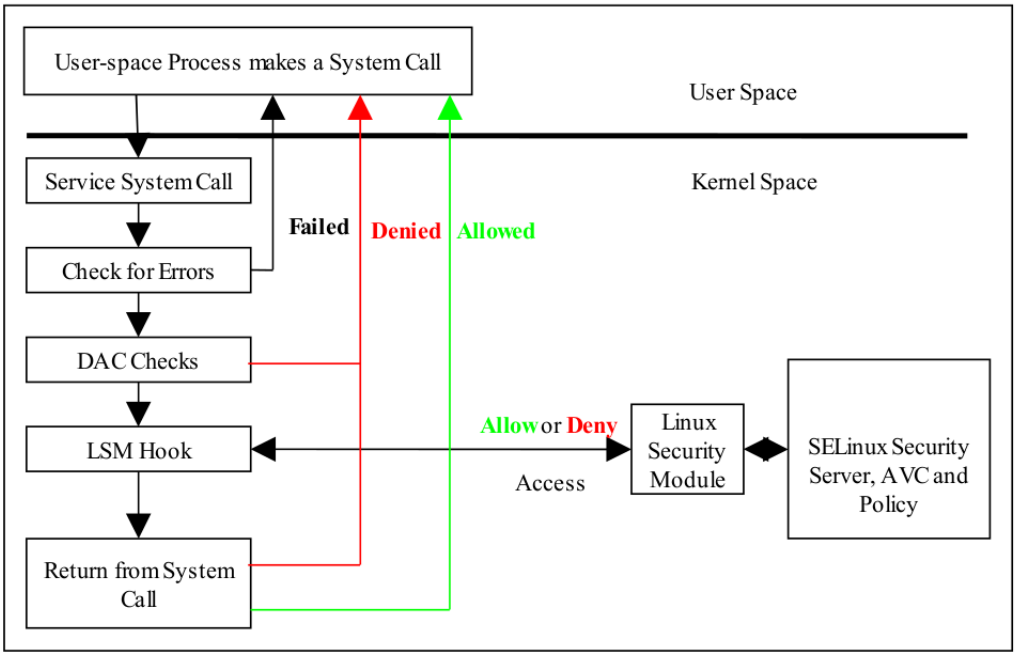

Fig. 2.1: Processing a System Call. [18]

SELinux determines access based on the **SELinux policy**. The policy is the set of rules in which interactions between subjects (processes) and objects (resources) are defined. Based on the configured security policies, the SELinux Security server decides if the subject has permission to perform the requested action on the object. The Security server needs the security context of subjects and objects, to decide if access is allowed or not [19].

SELinux context In SELinux Policy, all subjects and objects have defined special security labels called SELinux contexts. SELinux contexts are used by the security server to decide to grant access [20]. SELinux security context is shown as string which defines:

- SELinux user - defined with an "**u**" suffix,
- SELinux role - defined with an "**r**" suffix,
- SELinux type - defined with an "**t**" suffix,
- Multi-Category Security (MCS) / Multi-Level Security(MLS) - deifned with a **sensitivity-category** pair [20]".

Tab. 2.3: SELinux context.

SELinux user	:	SELinux role	:	SELinux type	:	MCS / MLS
↓		↓		↓		↓
system_u	:	object_r	:	lib_t	:	s0

An example of security context is above, where SELinux user is system_u SELinux role is object_r, SELinux type is lib_t and sensitivity level is s0, which is the lowest sensitivity level, with no categories.

SELinux users are Linux users mapped to the SELinux policy. SELinux users have different privileges and are used for a stricter policy. More Linux users can be mapped into one SELinux user, but only one SELinux user can be assigned to one Linux user. When Linux users log in, they automatically assigned an SELinux user. Linux users, who are not mapped into SELinux users, have the ___default___ flag set, which is by default assigned to the unconfined_u SELinux user [21].

SELinux role are attributes of the Role-Based Access Control (RBAC) security model, which is used by SELinux. *"RBAC in the Role-Based Access Control (RBAC) model, access to resources is based on the role assigned to a user [22]."* Each SELinux user has one or more assigned roles, to which they can be authorized, but only one role can be active at a time [23]. Assigned SELinux roles to SELinux users in Red Hat Enterprise Linux and Fedora configurations are shown below:

Tab. 2.4: SELinux users authorized for roles.

	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

"User controls the reachable roles and the roles control the reachable types [24]."

SELinux type is used in SELinux policy to define in which domain a process is running and what type is associated with an object. Type identifier is the most used part of the SELinux context to determine access between subjects and objects [25].

Type enforcement (TE) is a concept based on subject-access-object. *"In SELinux, type enforcement is implemented based on the labels of the subjects and objects [25]."* If a process wants to access files it must be allowed in SELinux Policy rules [18].

SELinux Policy Rules are access control rules specifying what is allowed. Each rule has form:
<rule name> <source domain> <target type> : <object class> <permission> [18].

The following example shows the Allow rule used in SELinux policy, where httpd_t is the label for Apache[1] process and httpd_log_t is the label for files in /var/log/httpd file-context file.

Tab. 2.5: Allow rule.

allow	**httpd_t**	**httpd_log_t**	**:**	**file**	**read**
↑	↑	↑		↑	↑
Rule name	Source domain	Target type	:	Object Class	Permissions

This rule means: *"APACHE process can READ its LOGGING FILE"* [26].

Multi-Category Security Multi-Category Security is an enhancement to SELinux, where access to files is also restricted by the categories assigned to users and files [28].

Multi-Level Security Multi-Level Security manages access between processes and resources based on four levels of security:

- Top secret,
- Secret,
- Confidential,
- Unclassified.

Subjects with a defined level of security can conduct operations on objects with different security levels. For example, a subject with a security level secret can write to files with security level top secret but cannot read from them [28].

[1] *"Apache HTTP Server is an open-source and free web server software [27]."*

SELinux policy is using 3 **source files**:

- **Type Enforcement (.TE) File** - Policy rules and transitions associated with the domain are written in type enforcement file. An example of policy rule defined in apache.te file:
  allow httpd_t httpd_log_t file read.
- **File Context (.FC) File** - File which defines security context to the paths for directories and files associated with domain. An example of type httpd_exec_t, assigned to a file declared in apache.fc file:
  /usr/sbin/apache - - gen_context(system_u:object_r:httpd_exec_t,s0).
- **Interface (.IF) File** - In these files are defined interface macros, that declare how a domain can be managed by other domains and these macros can be called by other modules. An example of interface, which is used to *"Allow the specified domain to read Apache log files [29],"* defined in apache.if file:
  apache_read_log(domain).

SELinux policy modules From this three files is compiled SELinux policy package, with suffix **.pp**, also called module. Module is compiled with command [30]:

Listing 2.1: SELinux policy module compilation.

```
# make -f /usr/share/selinux/devel/include/Makefile apache.pp
```

SELinux Policy modules are loaded through **semodule**, tool for managing SELinux policy modules:

Listing 2.2: SELinux policy module loading.

```
# semodule -i /path/to/apache.pp
```

Individual policy modules contain rules associated with applications. Except individual policy modules, there exists a base policy, which always needs to be loaded [31].

SELinux policy uses m4 and CIL language. A policy is enforced when SELinux is running in Enforcing mode. Enabled SELinux has two modes:

- **Enforced** - security policy is enforced,
- **Permissive** - security policy is not enforced but SELinux is logging denials [18].

When SELinux denies access, it is logged in /var/log/audit/audit.log, and can be accessed with the **Ausearch** tool that can search the audit daemon log :

Listing 2.3: Search the audit daemon log files with Ausearch.

```
# ausearch -m AVC -ts recent
```

Ausearch is used to search for **-m** message **AVC** which is Access Vector Cache, cache with SELinux decisions in the last 10 minutes defined by **-ts recent** parameter. If SELinux is disabled, policy cannot be enforced and it is not logging any denials [32].

For quick and simple SELinux configuration, parts of SELinux Policy are **booleans**. They can customize SELinux policy by enabling or disabling them.

Booleans can be managed with semanage-boolean. The **semanage** tool is used for managing SELinux policy. Command for listing all SELinux booleans and their state:

Listing 2.4: List SELinux booleans.

```
# semanage boolean -l
```

Command for changing state of boolean:

Listing 2.5: Manage SELinux booleans.

```
# semanage boolean -m --on deny_execmem
```

Where **-m** is a parameter for modify, **on/off** for enabling or disabling boolean, and **deny_execmem** is the name of the boolean [32].

SELinux has three types of policies:

- **minimum** - only the base policy, loadable modules are not compiled or loaded, all domains are unconfined, it can be used on low memory machine platforms,
- **targeted** - in this policy targeted processes are confined, primary mechanism of access control is type enforcement,
- **multi level security** - combines Type enforcement model with Bell-La Padula, where all subjects and objects are labeled with security label, only for servers [18].

Targeted policy is used in Red Hat Enterprise Linux based operating systems [33].

3 Red Hat Enterprise Linux based OS

This chapter describes Red Hat Enterprise Linux based operating systems.

3.1 Fedora Linux

Fedora Linux is an open source operating system built by the community. The Fedora community provides free, stable and robust software in Fedora distribution. The large community, and support from Red Hat, make Fedora distribution a very innovative platform. Fedora is released every 6 months, and has a short life cycle, usually 13 months. Red Hat is Fedora's primary sponsor. Fedora Linux is the upstream[1] of the Red Hat Enterprise Linux [34].

3.2 Red Hat Enterprise Linux

Red Hat Enterprise Linux (RHEL) is a commercial open source operating system. Red Hat supports the Fedora Project and other open source projects by contributing to the code. Then from these upstream projects, Red Hat makes stable and certified products and services with long-term support. Red Hat Enterprise Linux releases happen every few years and it is long-term supported. Security, reliability, and performance are main focuses of Red Hat Enterprise Linux [35].

3.3 CentOS

The free community version of Red Hat Enterprise Linux is CentOS. It is based on publicly available source code of Red Hat Enterprise Linux. Red Hat Enterprise Linux packages were modified to remove branding and artwork. CentOS provides the stability of Red Hat Enterprise Linux, but it is supported by the CentOS community and not by Red Hat. CentOS is supported for 10 ten years [36].

From CentOS was created CentOS Stream, which is the midstream between RHEL and Fedora. When Red Hat splits off from Fedora a new version of RHEL, all work will go into CentOS Stream. CentOS Stream will be upstream of the new RHEL version [37].

Fedora provides a lot of new features and new versions of software, RHEL provides supported and certified platforms and CentOS is a free community version of RHEL. All these operating systems have enabled by default SELinux policy.

[1] *"The upstream of a program or set of programs is the project that develops those programs [38]."*

SELinux was first introduced in RHEL 4, CentOS 4 and Fedora Core 2, and since then, is by default enabled and enforced [39] [40] [41].

4 Significant weaknesses over last 5 years

Red Hat Product Security provides updated security data. In one of the reports Common Vulnerabilities and Exposures are mapped to associated Common Weakness Enumerations[1], which are updated twice a day. From 1.1.2015 to 15.10.2019 it contained 8278 CVEs related to CWEs. The most commonly occurring weaknesses are displayed in the bar chart below, where the vertical axis shows the CWE-ID and bars represent percentage of their occurrence:

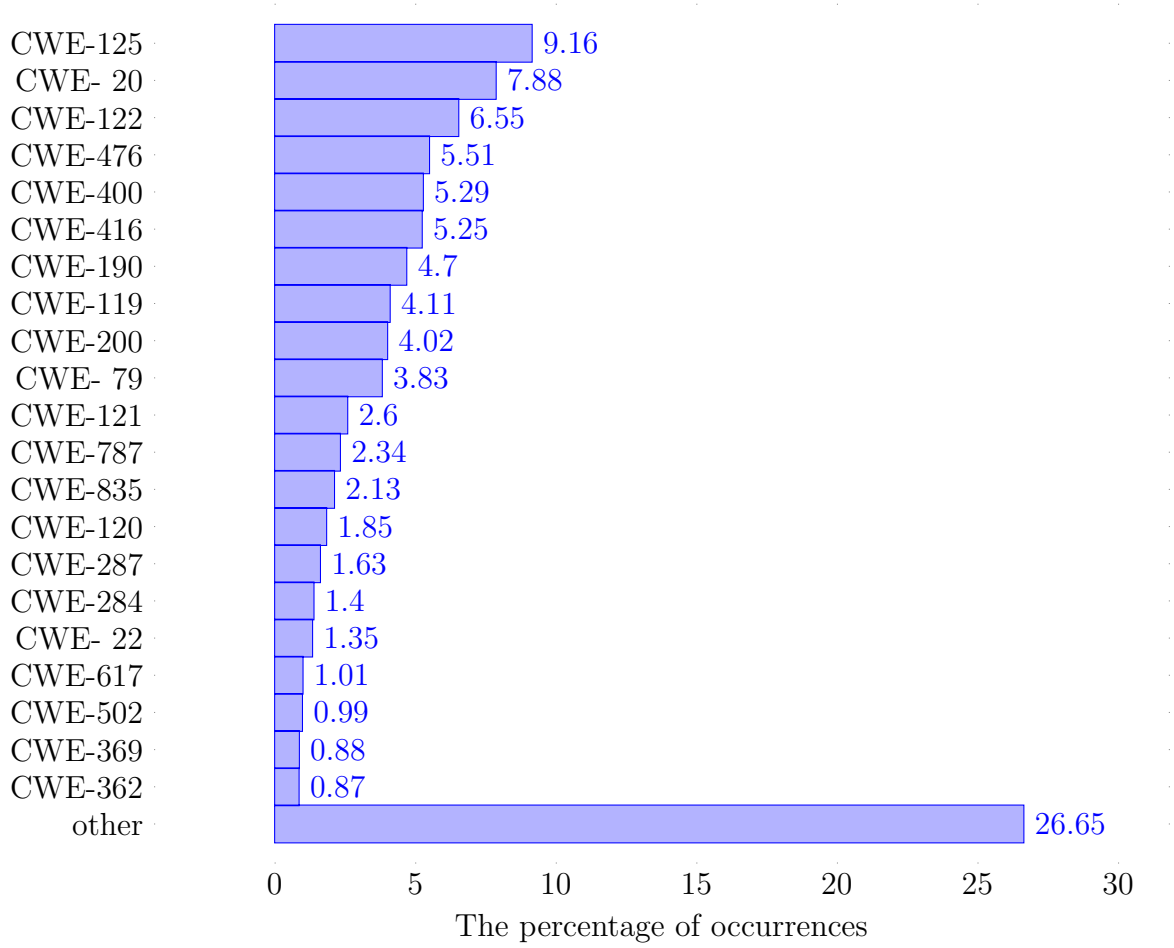

Fig. 4.1: The most commonly occurring weaknesses.

[1] https://www.redhat.com/security/data/metrics/cvemapcwe.txt

The name and the percentage of occurrences of each CWE-ID is in the table below.

Tab. 4.1: The most commonly occurring weaknesses (%).

CWE ID	CWE description	occurrence (%)
20	Improper Input Validation	7.88
22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	1.35
79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	3.83
119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.11
120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	1.85
121	Stack-based Buffer Overflow	2.6
122	Heap-based Buffer Overflow	6.55
125	Out-of-bounds Read	9.16
190	Integer Overflow or Wraparound	4.70
200	Information Exposure	4.02
284	Improper Access Control	1.40
287	Improper Authentication	1.63
362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	0.87
369	Divide By Zero	0.88
400	Uncontrolled Resource Consumption	5.29
416	Use After Free	5.25
476	NULL Pointer Dereference	5.51
502	Deserialization of Untrusted Data	0.99
617	Reachable Assertion	1.01
787	Out-of-bounds Write	2.34
835	Loop with Unreachable Exit Condition ('Infinite Loop')	1.9

Successful exploitation of weaknesses can lead to one or more technical impacts. For example, weakness CWE-20 (Improper Input Validation) occurs when: *"The product does not validate or incorrectly validates input that can affect the control flow or data flow of a program [42]."* Exploiting vulnerabilities associated with this weakness can result in denial of service, resource consumption, code execution, and

reading or modification of data [42].

Technical impacts of each weakness, with description and other information, are available on the MITRE website[2].

This database, with the database from Red Hat, where CVE is mapped into CWE, provides information about the occurrence of each technical impact.

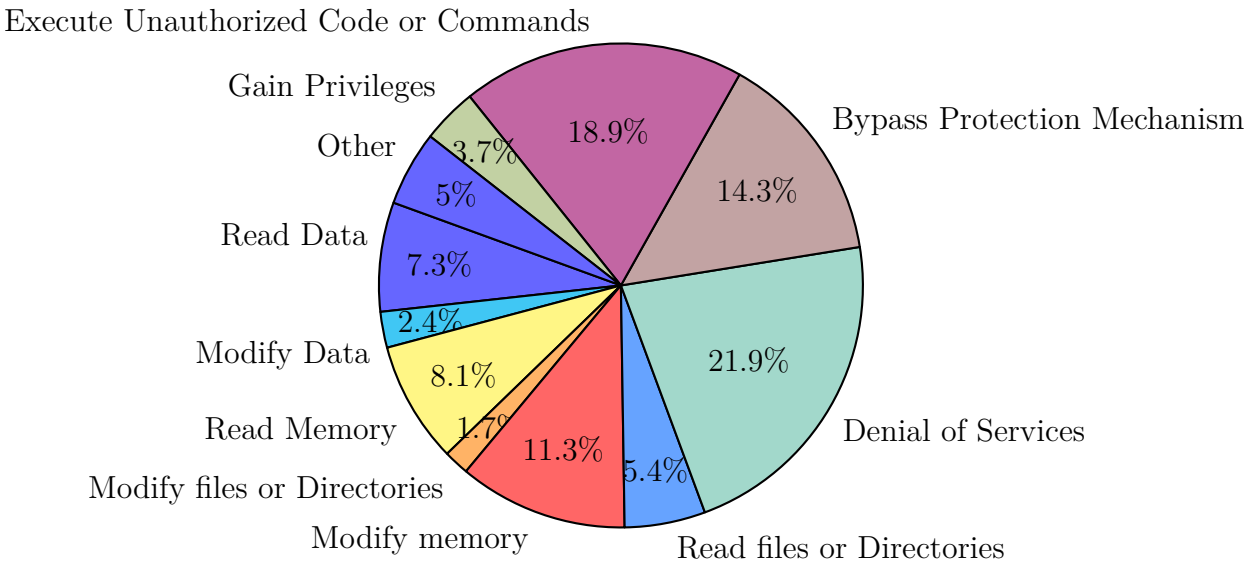

Fig. 4.2: The percentage of occurrences of each technical impact.

The graph shows that the most occurring technical impacts are denials of service. This is because the most numerous are weaknesses that manipulate memory, resources, or inputs, which cause a program to crash or consume resources. Memory manipulation can lead to other technical impacts. One of them, that often occurred is Execute Unauthorized Code or Commands. It can be used for privilege escalation, stealing data or controlling systems. Another impact that can be caused by memory manipulation is Bypass Protection Mechanism. Gain Privileges are not so often seen, but with escalated privileges, the attacker gains more permissions and can do more attacks. Reading and modification of data can be successfully mitigated with enabled SELinux and correct DAC permissions.

SELinux technology can mitigate the impact of exploited vulnerabilities and reduce security risks.

[2]https://cwe.mitre.org/data/definitions/1000.html

5 SELinux configuration design to mitigate technical impacts

SELinux is a flexible and powerful technology, which can be easily configured to mitigate the technical impacts of many weaknesses and vulnerabilities. SELinux is shipped with a default configuration that does not require ordinary user intervention. However, usability comes at the expense of security. SELinux has a lot of useful security features and protections that can be used to harden security, without limitations.

5.1 Memory Protection

In 2006, Uli Drepper added several SELinux memory checks:

- *execmem: make executable an anonymous mapping or private file mapping that is writable,*
- *execheap: make the heap executable,*
- *execstack: make the main process stack executable,*
- *execmod: make executable a file that has been modified by copy-on-write [18].*

These memory checks allow for the blocking of known attacks, like buffer overflows and code errors, that could lead to privilege escalations. If the process requires one of the listed permissions, it is probably not good coding style, and should be reported to application maintainers [43].

SELinux has several booleans to prevent execution of memory, stack, and heap:

Tab. 5.1: Executable memory booleans.

SELinux boolean name	SELinux boolean description
deny_execmem	*" Deny user domains applications to map a memory region as both executable and writable [44] "*
<domain>_execmem	Allow <domain> to use executable memory and executable stack

Tab. 5.2: SELinux users executable memory booleans.

SELinux boolean name	SELinux boolean description
selinuxuser_execmod	*"Allow all unconfined executables to use libraries requiring text relocation that are not labeled textrel_shlib_t [45]."*
selinuxuser_execheap	*"Allow unconfined executables to make their heap memory executable. Doing this is a really bad idea. Probably indicates a badly coded executable, but could indicate an attack. This executable should be reported in bugzilla [45]."*
selinuxuser_execstack	*"Allow unconfined executables to make their stack executable. This should never, ever be necessary. Probably indicates a badly coded executable, but could indicate an attack. This executable should be reported in bugzilla [45]."*

Because of the many bugs in applications, this permissions are by default allowed, to prevent SELinux denials [43].

To prevent processes from reading the memory of other processes, the boolean **deny_ptrace** was created.

Tab. 5.3: Deny process trace boolean.

SELinux boolean name	SELinux boolean description
deny_ptrace	*"Deny any process from ptracing or debugging any other processes [46]."*

"Ptrace, is system call where one process (the "tracer") may observe and control the execution of another process (the "tracee"), and examine and change the tracee's memory and registers [47]". It is used by debuggers, by tracing tools, and by specialized programs, to patch running programs. It allows one process to attack another process, or to read memory. Deny_ptrace can prevent privilege escalations and is recommended by Dan Walsh, if there is no need for debugging the applications [48].

5.2 Prevent processes from sharing files

In SELinux policy, file sharing services must explicitly enable SELinux booleans, if they want to share files or directories [49]. **Samba** is using the file and printer sharing protocol. To share files via Samba, files have to be labeled as samba_share_t. In the SELinux policy for Samba booleans are defined , which allow sharing all files in home directories or all filesystem files or Network File System files [50].

Tab. 5.4: Samba booleans.

SELinux boolean name	SELinux boolean description
smbd_anon_write	*"Allow samba to modify public files used for public file transfer services. Files/Directories must be labeled public_content_rw_t [51]."*
samba_enable_home_dirs	*"Allow samba to share users home directories [51]."*
samba_export_all_rw	*"Allow samba to export all read/write [51]."*
samba_export_all_ro	*"Allow samba to export all read only [51]."*
samba_share_fusefs	*"Allow samba to export ntfs/fusefs volumes [51]."*
samba_share_nfs	*"Allow samba to export NFS volumes [51]."*

Network File System (NFS) is the protocol for sharing and storing files and directories, over the network. To share any read-only files and directories, or all files and directories via NFS, certain booleans have to be turned on.

Tab. 5.5: NFS booleans.

SELinux boolean name	SELinux boolean description
nfsd_anon_write	*"Allow nfs servers to modify public files used for public file transfer services. Files/Directories must be labeled public_content_rw_t [52]."*
nfs_export_all_rw	*"Allow nfs to export all read/write [52]."*
nfs_export_all_ro	*"Allow nfs to export all read only [52]."*

Another way to store data over the network is **GlusterFS**. *"GlusterFS is a scalable network filesystem suitable for data-intensive tasks such as cloud storage and media streaming [53]."* Gluster services also require SELinux to be enabled, to share read-only files and directories, or all files and directories.

These booleans should be off by default to prevent misconfiguration and risk of data leakage. If the user doesn't want to change the state of certain booleans, he has to comment out a specific boolean, or delete it. The configuration of SELinux booleans is in the appendix B.1.

Tab. 5.6: Gluster booleans.

SELinux boolean name	SELinux boolean description
gluster_anon_write	*"Allow glusterfsd to modify public files used for public file transfer services. Files/Directories must be labeled public_content_rw_t [54]".*
gluster_export_all_rw	*"Allow glusterfsd to share any file/directory read/write [54]".*
gluster_export_all_ro	*"Allow glusterfsd to share any file/directory read only [54]".*

5.3 Prevent processes from connecting to port

To prevent unauthorized connection or transfer of information, all booleans which allow domains to connect to any port will be disabled. If some process requires port connection, the certain boolean has to be enabled in the configuration. The configuration of SELinux booleans is in the appendix C.1.

In the following table are listed booleans that manages mysql connections:

Tab. 5.7: Mysql booleans.

SELinux boolean name	SELinux boolean description
mysql_connect_any	*"Allow mysqld to connect to all ports [55]."*
mysql_connect_http	*"Allow mysqld to connect to http port [55]."*

5.4 Confining users

Linux users, who are not mapped into SELinux users, are by default assigned to the SELinux user unconfined_u, which is the least restrictive user. By mapping users to SELinux policy, the configuration would be more strict. User access is controlled with defined SELinux users:

- unconfined - unconfined user role,
- sysadm - general system administration role,
- staff - administrator's unprivileged user,
- user - generic unprivileged user,
- guest - least privileged terminal user,
- xguest - least privileged X user [18].

SELinux users and roles has different privileges as is it shown in table below.

Tab. 5.8: Permissions of confined users.

action	command	sysadm	staff	user	guest	xguest
list installed packages	$ rpm -qa	✓	✓	✓	✓	✓
look into systemd journal	$ journalctl –user	✓	✓	✓		
manage SELinux configuration	$ semanage	✓				
restart a service	$ systemctl	✓	✓			
make SSH connection	$ ssh	✓	✓			
run su	$ su -	✓				
run sudo	$ sudo	✓				
install new packages	$ dnf install	✓				
check for packages updates	$ dnf check-update	✓	✓	✓		
list your processes	$ ps -efZ	✓	✓	✓		
search for SELinux denials	$ ausearch -m avc	✓				
correct SELinux labels on your files	$ restorecon -Rv $HOME	✓				
list SELinux booleans	$ getsebool -a	✓	✓	✓		
get detailed info about SELinux	$ seinfo	✓	✓			
get basic info SELinux	$ sestatus	✓	✓	✓		
change your password	$ passwd	✓	✓			
ping	$ ping	✓	✓	✓		

SELinux users and roles are defined in section 2.3. Linux users can be mapped into SELinux users via the following command:

Listing 5.1: Map Linux user to the SELinux user.

```
# semanage login -a -s <selinux_user> <linux_user>
```

5.5 Disable unconfined and permissive domains

Unconfined processes run in an unconfined domain. They are minimally restricted, by SELinux, and have rights to almost all access. *"If an unconfined process is compromised, SELinux does not prevent an attacker from gaining access to system resources and data, but of course, DAC rules are still used [56]."* Unrestricted are also domains that run in permissive mode. Unlike unconfined domains, in this case SELinux logs accesses which would be denied [57].

In the following table are listed modules that manage unconfined and permissive domains:

Tab. 5.9: Unconfined and permissive modules.

SELinux module name	SELinux module description
unconfined	*"Unconfined domains run with no or few restrictions but do not log any requested access [57]."*
permissivedomains	*"Permissive domains are individual domains that are specified to run in permissive mode and log access what would be denied [57]."*

5.6 System lockdown

The last part of hardening SELinux is system lockdown. SELinux has 3 powerful booleans, that disallow processes from transitioning to privileged user domains, prevent confined users from inserting Linux Kernel modules (e.g SELinux policy module) and managing SELinux policy [58]. Booleans are listed in the following table:

Tab. 5.10: Secure mode booleans.

SELinux boolean name	SELinux boolean description
secure_mode	*"Do not allow transition to sysadm_t, sudo and su effected [45]."*
secure_mode_insmod	*"Do not allow any processes to load kernel modules [45]."*
secure_mode_policyload	*"Do not allow any processes to modify kernel SELinux policy [45]."*

When the last boolean, secure_mode_policyload, is enabled, users can only disable SELinux policy, during booting, by editing boot configuration.

5.7 Add Revert option

To allow users to return SELinux configuration to a previous state, in the local environment, the revert option is necessary. For this reason, local SELinux modifications will be extracted and saved. Local SELinux modifications can be extracted with the semanage tool:

Listing 5.2: Export SELinux modifications.

```
# semanage -o <output_file>                                    1
```

Import of saved SELinux modifications is also possible with semanage tool:

Listing 5.3: Import SELinux modifications.

```
# semanage -i <imput_file>                                     1
```

5.8 Fixed bug in the Flatpak SELinux policy

When disabling unconfined modules, with the semodule tool, there was an error:

Listing 5.4: Error in the Flatpak SELinux policy.

```
# semodule -d unconfined                                       1
Failed to resolve typeattributeset statement at /var/lib/     2
   selinux/targeted/tmp/modules/200/flatpak/cil:29
semodule:  Failed!                                             3
```

The problem was in the SELinux policy shipped with the Flatpak service. In the SELinux policy flatpak.te file there was an statement:

Listing 5.5: Statement in the flatpak.te file.

```
unconfined_domain(flatpak_helper_t)                            1
```

Unconfined_domain is an interface defined in an unconfined.if file and it is used to make the SELinux domain unconfined [59].

Modules that are not part of the base SELinux policy and could be removed, like unconfined modules, have to be in an optional policy block. *"The optional statement*

is used to indicate what policy statements may or may not be present in the final compiled policy [60].” because when they are removed, the SELinux policy will fail.

In the SELinux policy flatpak.te file, interface unconfined_domain wasn't in an optional block, so it was necessary to fix it by adding this optional_policy statement:

Listing 5.6: Add optional block in the flatpak.te file.

```
optional_policy(‘
    unconfined_domain(flatpak_helper_t)
’)
```

With a fixed flatpak.te file, a pull request[1] to Flatpak GitHub repository was made.

[1] #3639: Add unconfined_domain(flatpak_helper_t) to an optional_policy block. https://github.com/flatpak/flatpak/pull/3639

6 Automation tool Ansible

Ansible is an automation tool that provides management of multiple hosts, nodes with SSH. Ansible has to be installed on one control node, which manages other hosts. It uses a playbook, which contains a set of instructions, also called tasks, applied on hosts. The sequence of tasks is a play. Each playbook can contain one or more plays. Hosts are defined in an inventory. It contains information about addresses, IP addresses, groups, credentials, etc. Managed hosts don't need Ansible installed [61].

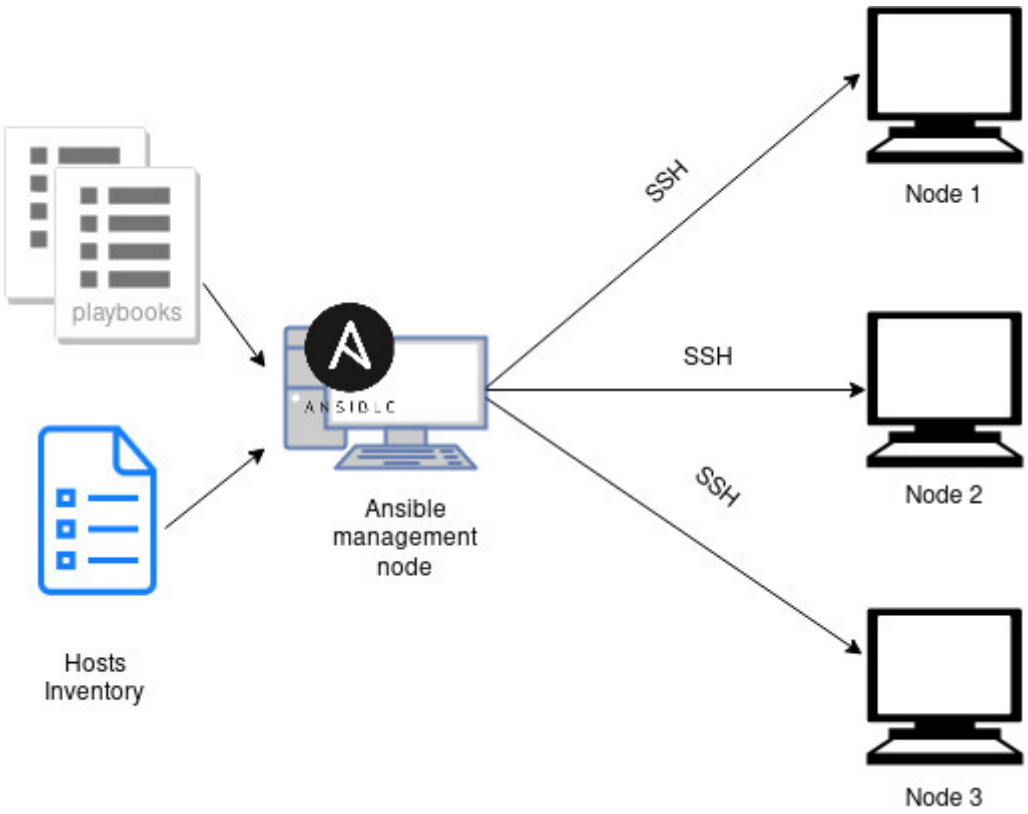

Fig. 6.1: Automation tool Ansible.

A framework with variables, tasks, files and modules is called Ansible Role. *"Roles are ways of automatically loading certain vars_files, tasks, and handlers based on a known file structure [62]."*

For simple system configuration Linux System Roles, were created. These are collections of Ansible roles, for managing system configurations. Fully supported roles are:

- selinux,

- kdump,
- network,
- timesync,
- postfix [63].

An SELinux role provides a configuration interface that can easily manage SELinux state, SELinux booleans state, file contexts, logins and port [64].

7 SELinux configuration in Ansible

When an Ansible play is run, Ansible informs about the executed play, which is Hardening SELinux, in this case. After that, it will gather information about hosts and their IP addresses from hosts. Then the SELinux role is loaded, which provides the SELinux configuration interface. Thereafter, local SELinux modifications are extracted to the file .defaultconf. Next, unconfined module and permissive domains are disabled. After that, it sequentially executes all defined tasks. Booleans configuration is based on files cant_connect.yml, deny_execmem.yml and deny_export.yml, which denies the domains connecting to the port, export files or execute the memory/ heap/ stack. In the next step, Linux users are mapped into the SELinux policy. At the end is a recapitulation of the executed play, succeeded and failed tasks.

The flowchart represents the steps of an Ansible configuration, in sequential order.

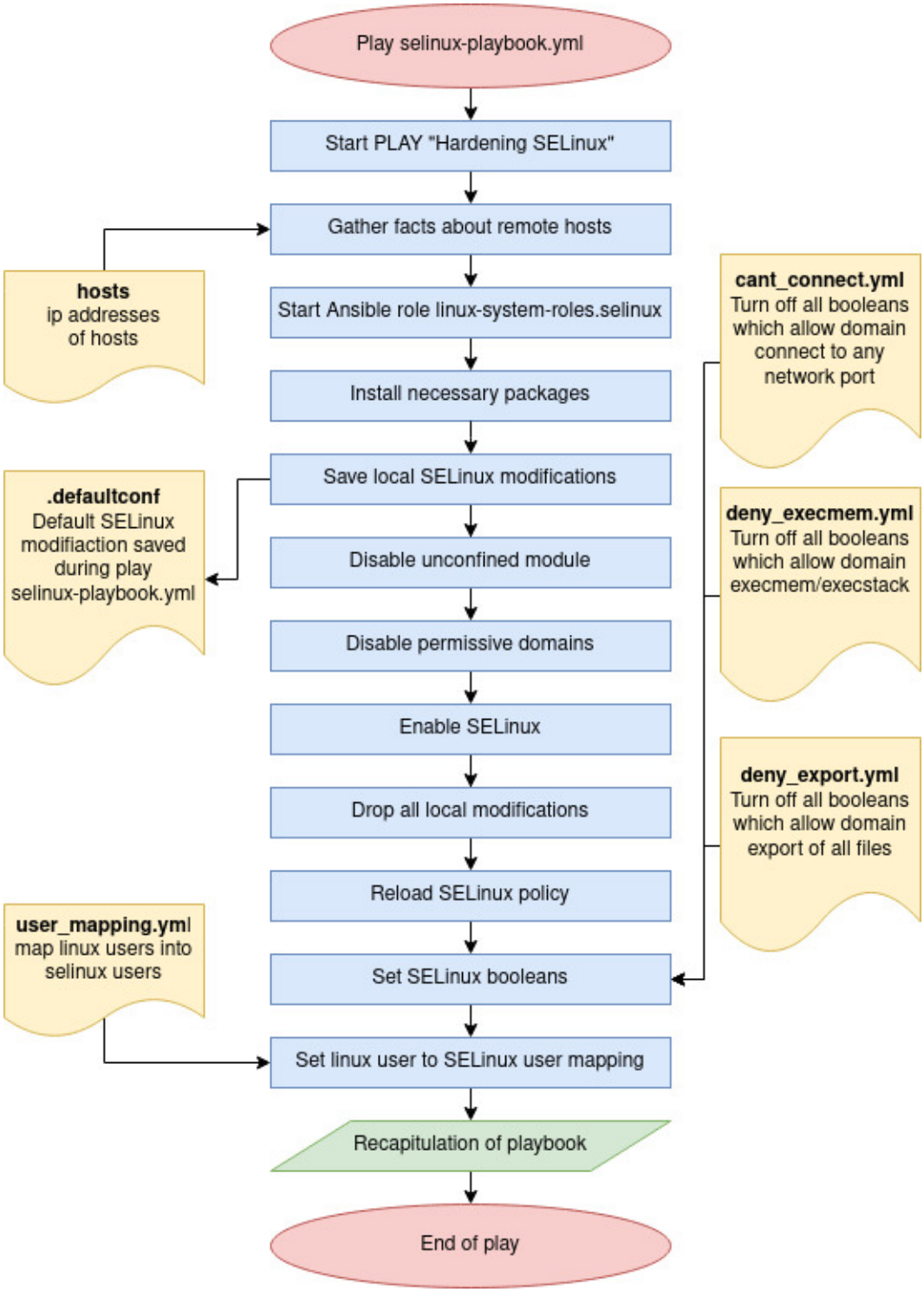

Fig. 7.1: Ansible flowchart.

The Ansible tool has to be installed to configure and apply playbooks. For SELinux configuration, Ansible also needs the SELinux role to be installed. All necessary packages are installed by following commands:

Listing 7.1: Ansible installation.

```
# dnf install ansible                                          1
# ansible-galaxy install linux-system-roles.selinux            2
```

The actual state of configuration is available on Gitlab and can be downloaded by:

Listing 7.2: Download SELinux configuration in the Ansible.

```
$ git clone https://gitlab.com/5umm3r15/selinux-hardening.git   1
```

The downloaded repository has a structure:

```
selinux-hardening
├── files
│   ├── cant_connect.yml
│   ├── deny_execmem.yml
│   └── deny_export.yml
├── inventory
│   └── hosts.yaml
├── revert
│   ├── revert-playbook.yml
│   └── defaultconf
├── selinux-playbook.yml
├── system-lockdown
│   ├── secure_mode.yml
│   └── selinux-lockdown-playbook.yml
└── users
    └── user_mapping.yml
```

Fig. 7.2: Repository structure.

7.1 The main configuration

The main SELinux configuration in Ansible is managed in the selinux-playbook.yml file. It includes a file with hosts, tasks, SELinux role, files to manage SELinux booleans and the file where users are mapped into SELinux users.

Listing 7.3: Main configuration file selinux playbook.yml.

```
# selinux-playbook.yml
---
- name: "Hardening SELinux"
  hosts: localhost
  become: true
  become_method: sudo
  become_user: root

  pre_tasks:
    - name: install policycoreutils-python-utils
      package:
        name: policycoreutils-python-utils
        state: latest
      when: ansible_distribution == 'Fedora'

    - name: install policycoreutils-python
      package:
        name: policycoreutils-python
        state: latest
      when: ansible_distribution == 'CentOS' or
         ansible_distribution == 'Red Hat Enterprise Linux'

    - name: save configuration
      shell: semanage -o ~/.defaultconf
      ignore_errors: yes

    - name: disable unconfined module
      shell: semodule -d unconfined
      ignore_errors: yes

    - name: disable permissive domains
      shell: semodule -d permissivedomains
      ignore_errors: yes
```

```
  vars_files:                                                          34
    # booleans hardening                                               35
    - files/cant_connect.yml                                           36
    - files/deny_execmem.yml                                           37
    - files/deny_export.yml                                            38
    # users mapping                                                    39
    - users/user_mapping.yml                                           40
                                                                       41
  vars:                                                                42
    selinux_policy: targeted                                           43
    selinux_state: enforcing                                           44
    selinux_booleans: "{{ cant_connect }} + {{ deny_execmem }}         45
        + {{ deny_export }}"
    selinux_logins: "{{ user_mapping }}"                               46
                                                                       47
  roles:                                                               48
    - linux-system-roles.selinux                                       49
```

Currently, a configuration is targeted on the localhost. But in the directory inventory is the file hosts.yaml, in which can be defined hosts. *"Ansible works against multiple managed nodes or "hosts" in your infrastructure at the same time, using a list or group of lists know as inventory [65]."* The defined group name is webserver and the IP address of the remote host.

Listing 7.4: The Ansible hosts file /inventory/hosts.

```
#/inventory/hosts                                                      1
[webservers]                                                           2
192.168.122.164                                                        3
```

When managing multiple hosts, it is necessary to define them to the "hosts" in the main configuration file selinux-playbook.yml and use the "-i inventory/hosts" option when running a playbook.

Pre_tasks used in configuration are tasks, which will be used executed before any other tasks. In the configuration are 5 pre_tasks:

- **policycoreutils-python-utils** - SELinux core policy utilities for Fedora
- **policycoreutils-python** - SELinux core policy utilities for CentOS and Red Hat Enterprise Linux
- **export default configuration** - exports custom SELinux modifications on system to file .defaultconf in hosts home directory - before configuration,

- **disable unconfined module** - with semodule command to remove unconfined processes,
- **disable permissive domain** - with semodule command to remove all permissive domains.

Linux users are mapped into SELinux policy in file the /users/user_mapping.yml. After *login* is written username, that will be confined by SELinux user. The SELinux user is defined after *seuser*. When state has the value present, the user is mapped into SELinux users, and when the value is absent, the user is removed from the policy.

Listing 7.5: SELinux user mapping file /users/user_mapping.yml.

```
# /users/user_mapping.yml                                                    1
# Usage:                                                                     2
#  - { login: '<username>', seuser: '<selinux_user>', state:                 3
   '<absent/present>'}
                                                                             4
user_mapping:                                                                5
  - { login: '__default__', seuser: 'staff_u', state: 'present               6
    ' }
```

The state of booleans for memory protection is configured in /files/execmem.yml. To allow domains execmem, certain booleans have to change values. To turn on, change *state* to *on*, to turn off, change state to *off*. Each boolean has a description in the SELinux man page.

Listing 7.6: Memory protection configuration file /files/deny_execmem.yml.

```
# /files/deny_execmem.yml                                                    1
# Turn off all booleans which allow domain execmem/execstack                 2
deny_execmem:                                                                3
# Determine whether boinc can execmem/execstack.                             4
      - { name: 'boinc_execmem', state: 'off', persistent: '                 5
        yes' }
```

The full configuration file is in an appendix A.1.

Network connections of domains are determined in ./files/cant_connect.yml. By default all domains have turned off booleans, which allow them to connect to network ports in this configuration. If some domain needs to connect to a port, the state of the relevant boolean has to be on. Each boolean has a description.

Listing 7.7: Network connections configuration file files/cant_connect.yml.

```
# files/cant_connect.yml
#Turn off all booleans which allow domain connect to any
   network port
cant_connect:

# Allow cluster administrative domains to connect to the
   network using TCP.
      - { name: 'cluster_can_network_connect', state: 'off',
        persistent: 'yes' }
```

The full configuration file is in an appendix C.1.

Files to share by Samba, GlusterFS or NFS are managed in /files/deny_export.yml If some files have to be shared, the related boolean needs to be turned on. Each boolean has a description.

Listing 7.8: Files sharing configuration file files/deny_export.yml.

```
# files/deny_export.yml
deny_export:
### Gluster ###

# Allow glusterfsd to modify public files used for public
   file transfer services. Files/Directories must be labeled
   public_content_rw_t.
      - { name: 'gluster_anon_write', state: 'off', persistent
        : 'yes' }
```

The full configuration file is in an appendix B.1.

After customization of the configuration files, the selinux-playbook can be run with:

Listing 7.9: Run Ansible playbook.

```
# cd selinux-hardening
# ansible-playbook selinux-playbook.yml
```

7.2 Revert option

To return the SELinux configuration to before Ansible configuration, the revert option was added in ./revert/revert-playbook.yml.

Listing 7.10: Revert configuration file revert/revert-playbook.yml.

```
# revert/revert.yml
---
- name: "Revert Hardening SELinux"
  hosts: local
  become: true
  become_method: sudo
  become_user: root

  vars:
    selinux_policy: targeted
    selinux_state: enforcing
    selinux_all_purge: true

  vars_prompt:
    - name: reboot
      prompt: "Reboot now? (yes/no)"
      private: no

  tasks:
  - name: import default configuration.
    shell: semanage -i ~/.defaultconf
    ignore_errors: true

  post_tasks:
    - name: Rebooting
      when: reboot == "yes"
      command: reboot

  roles:
      - linux-system-roles.selinux
```

Hosts are defined in the directory Inventory, in the file hosts.yaml, currently targeted on the localhost. Default configuration stored in .defaultconf file is applied

on hosts. When running this playbook, the user is asked if he wants to reboot the system. If yes, the system is rebooted afterward.

Listing 7.11: Run revert configuration playbook.

```
# ansible-playbook revert/revert-playbook.yml
```

7.3 System lockdown option

For system lockdown, there is another playbook, located in the directory system-lockdown. **This playbook should be executed only when the user knows what he is doing.**

Listing 7.12: System lockdown file system-lockdown/selinux-lockdown-playbook.yml.

```
# system-lockdown/selinux-lockdown-playbook.yml
---
- name: "Lockdown system with SELinux"
  hosts: localhost
  become: true
  become_method: sudo
  become_user: root

  vars:
    selinux_policy: targeted
    selinux_state: enforcing

  vars_files:
    - system-lockdown/secure_mode.yml

  roles:
    - linux-system-roles.selinux
```

Hosts are again targeted on the localhost and can be defined in the directory Inventory in the file hosts.yaml. The system lockdown playbook includes secure_mode booleans, declared in the secure_mode.yml file.

Listing 7.13: Secure_mode configuration file system-lockdown/secure_mode.yml.

```
# system-lockdown/secure_mode.yml
selinux_booleans:
    - { name: 'secure_mode', state: 'on', persistent: 'yes' }
    - { name: 'secure_mode_insmod', state: 'on', persistent: '
      yes' }
    - { name: 'secure_mode_policyload', state: 'on',
      persistent: 'yes' }
```

If this playbook is executed, with all secure booleans enabled, users will not be able to revert system lockdown, by executing the revert playbook. Moreover, users will not be able to perform any operations with SELinux.

Listing 7.14: Run system lockdown playbook.

```
# ansible-playbook system-lockdown/selinux-lockdown-playbook.
   yml
```

8 Security vs usability

8.1 Graphical login managers

Graphical login managers will crash when the boolean deny_execmem is enabled. Users, who want to use a graphical login manager, have to set the state of boolean deny_execmem to off. It can be easily done, in the file files/deny_execmem.yml, in the Ansible configuration, by changing the state of the boolean to off:

Listing 8.1: Disable SELinux boolean deny_execmem in the Ansible configuration.

```
{ name: deny_execmem, state: 'off', persistent: 'yes' }
```

Or use semanage tool:

Listing 8.2: Disable SELinux boolean deny_execmem with semanage.

```
# semanage boolean -m --off deny_execmem
```

8.2 Unconfined and confined users

Even when you disable an unconfined module, when unconfined users run processes that are not confined, it will run in an unconfined domain. This is why it is important to confine users into SELinux users. When a confined user wants to perform a privileged operation, he has to become a confined administrator [66].

Only 3 SELinux users can switch to the root user account with command su:

- unconfined_u,
- sysadm_u,
- staff_u.

However, user staff_u cannot do it directly, he has to change from the staff user domain to sysadm or unconfined user domain. This is possible only by switching roles with the tool newrole[1]. Newrole tool can be installed:

Listing 8.3: Install newrole tool.

```
$ sudo dnf install policycoreutils-newrole
```

Transition to the sysadm role can be done with command:

[1] newrole - run a shell with a new SELinux role

Listing 8.4: Switch SELinux role.

```
$ newrole -r sysadm_r
```

The SELinux context associated with a user can be displayed with the id -Z command:

Listing 8.5: Display SELinux context.

```
$ id -Z
staff_u:sysadm_r:sysadm_t:s0
```

The example command shows that the user staff_u is in a sysadm_r role and operates in sysadm_t user domain, so now he can execute su command.

Confined users can have managed privileges through booleans. In case of confining users, it is important to pay attention to the boolean secure_mode, which is described below.

8.3 Confined users and secure_mode boolean

When the secure_mode boolean is enabled, it protects confined users from transitions to sysadm domain or using su command to switch to the root user . In SELinux policy there is only one SELinux user, which can transite to the sysadm user domain. It is the SELinux user staff_u.

Whether the secure_mode boolean is enabled or disabled, unconfined users are allowed to get root privileges. On the other hand, SELinux user staff_u cannot execute su command, no matter what the state of the secure_mode boolean.

An example is demonstrated by the Linux user niki, mapped into the SELinux user staff_u, with an unsuccessful attempt to execute a sudo su command to become root:

Listing 8.6: Staff_u executes sudo. command.

```
[niki@localhost ~]$ id -Z
staff_u:staff_r:staff_t:s0
[niki@localhost ~]$ sudo su
[sudo] password for niki:
bash: /root/.bashrc: Permission denied
```

The diagram displays attempts to gain superuser privileges, based on the state of the SELinux boolean secure_mode and SELinux user.

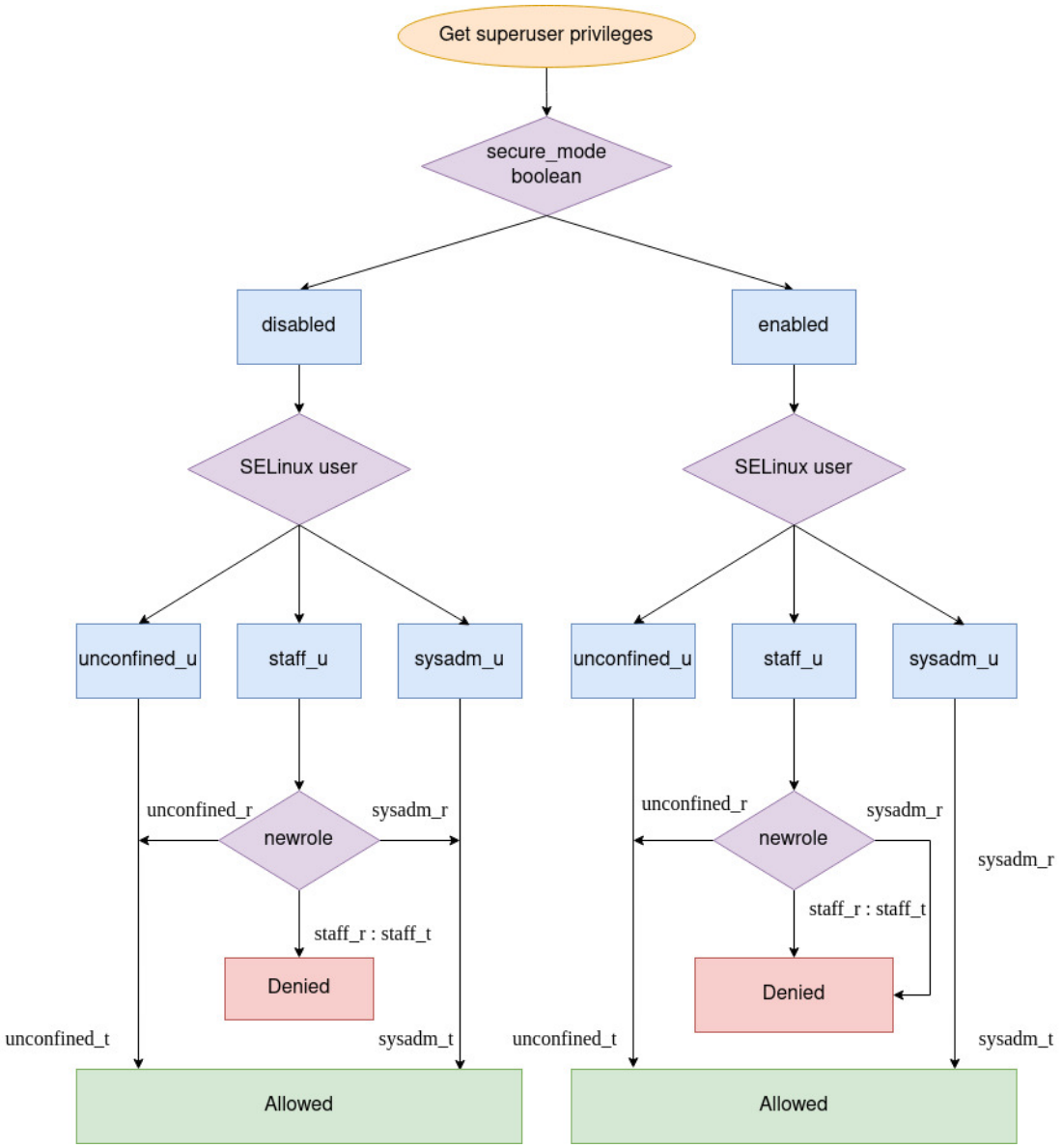

Fig. 8.1: Get superuser privileges.

As illustrated in the diagram, the state of the secure_mode boolean affects whether the SELinux user staff_u can switch the role to sysadm role or not. When the secure_mode boolean is disabled, the SELinux user staff_u is able to switch role, with the newrole command, to unconfined_r or sysadm_r roles. And then, he is allowed to execute su command. But, an enabled secure_mode boolean doesn't allow SELinux staff user to change role to sysadm, as it is shown on following example:

Listing 8.7: Switch SELinux role to sysadm.

```
[niki@localhost ~]$ newrole -r sysadm_r
Password:
failed to exec shell
: Permission denied
[niki@localhost ~]$ id -Z
staff_u:staff_r:staff_t:s0
```

However, in case of switching from staff to an unconfined role, the state of the secure_mode boolean is not important. The SELinux user staff is still able to transit to an unconfined role, and use the su command to switch to the root user , in an unconfined domain, even when the secure_mode boolean is enabled.

In this example, The Linux user niki, mapped into SElinux user staff, changes role to unconfined role and successfully switches to the root user , no matter what the state of the secure_mode boolean:

Listing 8.8: Switch SELinux role to unconfined.

```
[niki@localhost ~]$ id -Z
staff_u:staff_r:staff_t:s0
[niki@localhost ~]$ newrole -r unconfined_r
Password:
[niki@localhost ~]$ id -Z
staff_u:unconfined_r:unconfined_t:s0
[niki@localhost ~]$ sudo su
[sudo] password for niki:
[root@localhost niki]#
```

Let's look deeply at this. The secure_mode boolean is defined in the selinuxutil.te file.

Listing 8.9: Description of the secure_mode boolean.

```
# if secure mode is enabled, then newrole
# can only transition to unprivileged users
if(secure_mode) {
        userdom_spec_domtrans_unpriv_users(newrole_t)
} else {
        userdom_spec_domtrans_all_users(newrole_t)
}
```

The char # is used for comments, which defines the boolean. There is a conditional statement which declares, that if the secure_mode boolean is enabled, then users can switch role through a new role application only to unprivileged users domains. If the boolean is disabled, then users can transit to all user domains, who are allowed in the default SELinux configuration.

With the seinfo tool, it is possible to find out which SELinux users are defined as unprivileged users:

Listing 8.10: Find unprivileged users.

```
$  seinfo -xaunpriv_userdomain

Type Attributes: 1
   attribute unpriv_userdomain;
    guest_t
    staff_t
    staff_wine_t
    unconfined_t
    user_t
    user_wine_t
    xguest_t
```

Between unprivileged users is also declared unconfined user, who is the least restrictive user. This can be a security issue, and it was reported to Red Hat Bugzilla[2], which is the Red Hat bug-tracking system.

This issue can be fixed by manually modifying SELinux roles, linked with the staff user, with the semanage command, and adding only staff_r, sysadm_r and system_r roles without unconfined_r:

Listing 8.11: Modify linked roles.

```
$ sudo semanage user -m -R "staff_r sysadm_r system_r" staff_u
```

And check it with:

Listing 8.12: List linked roles.

```
$ sudo semanage user -l | grep staff_u
staff_u  user   s0  s0-s0:c0.c1023  staff_r sysadm_r system_r
```

[2]BZ1840851: Secure_mode boolean allows staff SELinux user switch to unconfined. https://bugzilla.redhat.com/show_bug.cgi?id=1840851

With modification of the linked SELinux roles to the SELinux user staff_u, and enabling the secure_mode boolean, the only SELinux user who can gain root privileges is sysadm.

8.4 System lockdown with secure_mode_policyload boolean

When the secure_mode_policyload boolean is enabled, the system prevent management of SELinux policy, modules, booleans, and the state of SELinux.

In the example, an unsuccessful try to disable secure_mode_policyload boolean is shown:

Listing 8.13: Disable secure_mode_policyload boolean.

```
[root@localhost n]# semanage boolean -m --off                    1
   secure_mode_policyload
libsemanage.bool_commit_list: libselinux commit failed (         2
   Permission denied).
libsemanage.bool_commit_list: could not commit boolean list (    3
   Permission denied).
libsemanage.dbase_activedb_flush: could not flush active         4
   database (Permission denied).
libsemanage.semanage_commit_components: could not commit         5
   local/active modifications (Permission denied).
PermissionError: [Errno 13] Permission denied                    6
```

The only solution is to reboot the system, and change the SELinux state, during boot time, by setting one of these kernel parameters:

- enforcing=0 to run selinux in permissive mode,
- selinux=0 to disable selinux.

Boot configuration can be edited by pressing the letter e, during boot time, and adding this kernel parameter enforcing=0 to run selinux in permissive mode:

After this, the SELinux policy can be modified in userspace.

Fig. 8.2: Kernel parameter to change SELinux state.

8.5 Performance impact

In this section the performance of the system with enforcing + targeted SELinux is compared to SELinux disabled. The performance is compared on Fedora 30, installed on a virtual machine, using Kernel-based Virtual Machine (KVM), which is an open-source virtualization technology.

Systemd-analyze The first tested was system boot-up time with the systemd-analyze tool - analyze and debug system manager. A simple systemd-analyze command, without parameters, prints the time spent in the kernel, initial RAM disk, and time to initialize system userspace [67]. The systemd-analyze tool has a lot of options, for example, it can list units that took the most time during boot. However, these results can be misleading, because services that have been evaluated as the most critical units have only waited for other services (e.g. Plymouth-quit-wait.service). For this reason, the only results that have been used were from the systemd-analyze command, without parameters.

Systemd-analyze was tested 10 times with enforcing + targeted SELinux, and 10 times with disabled SELinux, on a virtual machine, and the results are represented in the bar graph. Values indicate the time to boot up system and blue columns are for SELinux enforcing + targeted and red columns are for SELinux disabled.

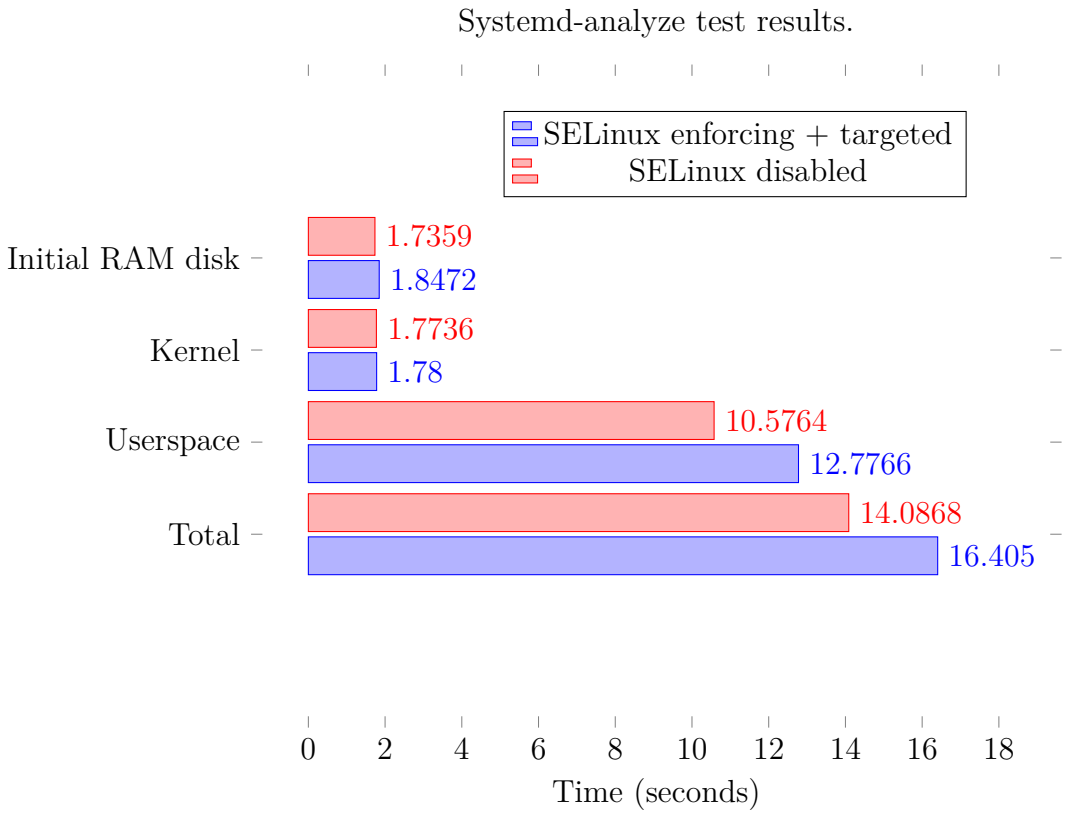

Fig. 8.3: System boot-up time.

The first two columns display time spent in the kernel, where disabling SELinux increased performance by about 0.36 %. The second two columns visualize the time spent in the initial RAM disk, with a difference of about 6.03 %. The next two bars represent time to reach userspace, and disabling SELinux decreased the time by about 17.22 %. Total time is illustrated by the last two columns where disabling SELinux boosted the performance by about 14.13 %.

UnixBench

Another commonly-used benchmark suite is UnixBench, that runs multiple tests and from the results indexes are constructed. *"These test results are then compared to the scores from a baseline system to produce an index value, which is generally easier to handle than the raw sores [68]."* The baseline system is a machine called George,[3] whose scores were set at 10.0. A higher index indicates higher performance.

For this purpose, the following were measured: copying of file with different total and block sizes; writing and reading from pipes; switching integers in a pipe; process creation; calls of the execl function; repeated execution of shell scripts; and finally entering and exiting of the operating system, with a system call.

[3]SPARCstation 20-61 with 128 MB RAM, a SPARC Storage Array, and Solaris 2.3

UnixBench runs tests with enforcing and targeted SELinux, and then with disabled SELinux, on a virtual machine, and results are represented in the bar graphs. Values indicate the indexes and blue columns are for SELinux enabled + enforcing and red columns for SELinux disabled.

The bars in the first graph display average indexes of file copy tests. File copy tests represent the data written from one file to another file, with different sizes of files and buffers. Six tests of file copying were performed, for each file size, and for each SELinux state. Each test runs 30 seconds and checks writing, reading, and copying functions. Measurements were in kilobytes per second.

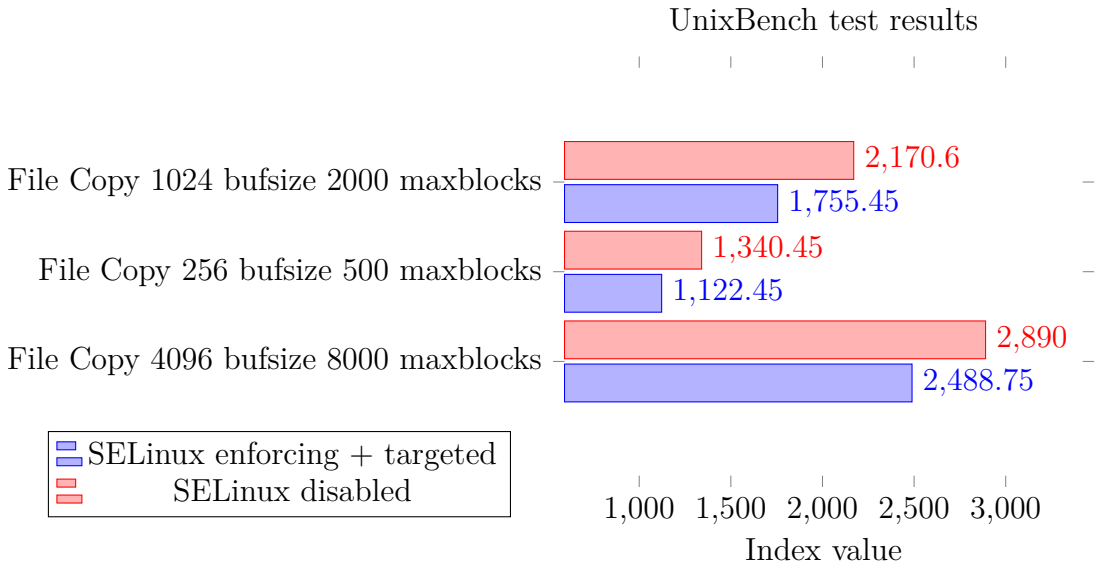

Fig. 8.4: File transfer performance.

Average index of copying 1 kilobyte file with buffer size 2000 bytes is increased by 19.12 % when SELinux is disabled. Index of copying 256 bytes file with buffer size 500 bytes is increased by 16.26 % and 4 kilobytes file with buffer size 8000 bytes is increased by 13.88 % by disabling SELinux.

Next graph show average indexes of Process creation and Shell scripts tests. Process creation test *"repeatedly calls the fork function to create a process and then immediately exits the process [69]."* Shell scripts measure the number of times the process can be created and execute a 1 script or set of 8 copies of a scripts repeatedly within 1 minute.

Shell scripts were performed for each SELinux enforcing + targeted and SELinux disabled.

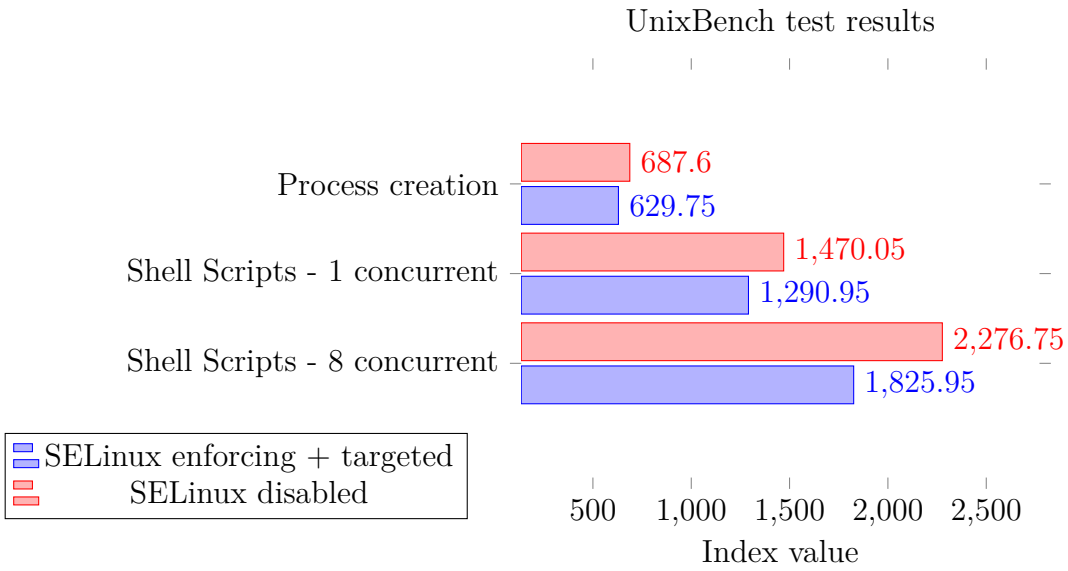

Fig. 8.5: Process creation and Shell scripts execution performance.

The average index of the process creation test was decreased by enforcing + targeted SELinux. For shell scripts two sets of six tests run for each number of executed scripts. The average index of creation and execution of one script was increased by 12.18 % with SELinux disabled. The index o f creation and execution of a set of eight copies of shell scripts were increased by 19.8 % with SELinux disabled.

The third graph illustrates system and execl calls. The system calls overhead *"test calculates the overhead for entering and exiting the operating system [69]"* with the execution of the waitpid function within 10 seconds. Excel is a test of the exec call function, where the number of process images replaced with new programs, per 10 seconds, is measured [70].

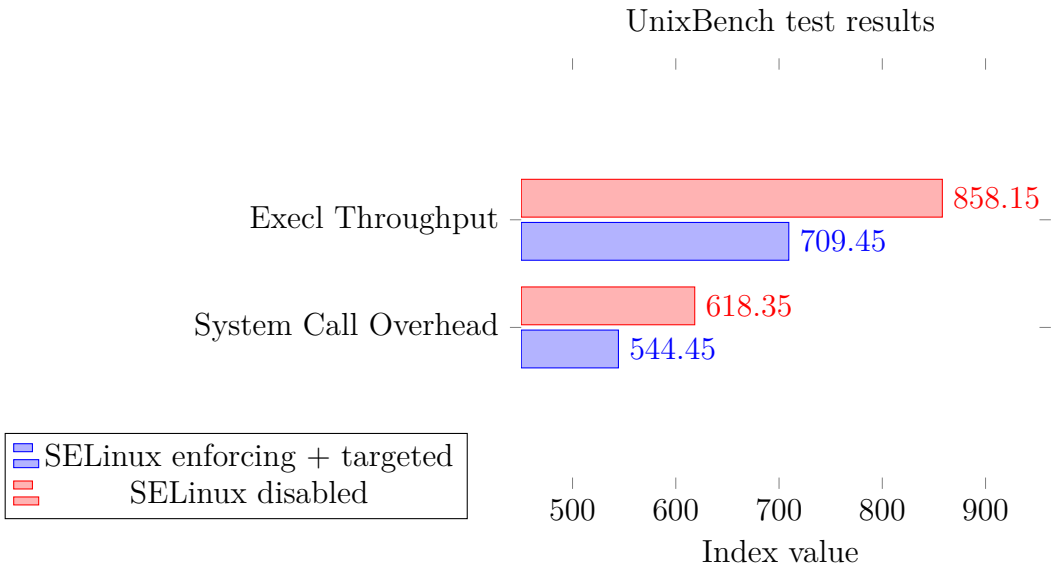

Fig. 8.6: System call performance.

Six tests for calling execl functions were running for each state of SELinux, and disabling SELinux raised the average index by 17.33 %. System call overhead tests ran twenty times for each enforcing and disabled SELinux. The index of disabled SELinux is higher by 11.95 %.

The last graph shows indexes of testing the speed of writing and reading from the pipeline. The pipe throughput *"test opens a pipeline, writes 512 bytes to the pipeline, and then reads the data from the pipeline [69]."* The number of operations is counted over 10 seconds. Pipe based context switching tests are described as: *" One process writes data to pipeline 1 and reads data from pipeline 2. The other process writes data to pipeline 2 and reads data from pipeline 2 [69]."*

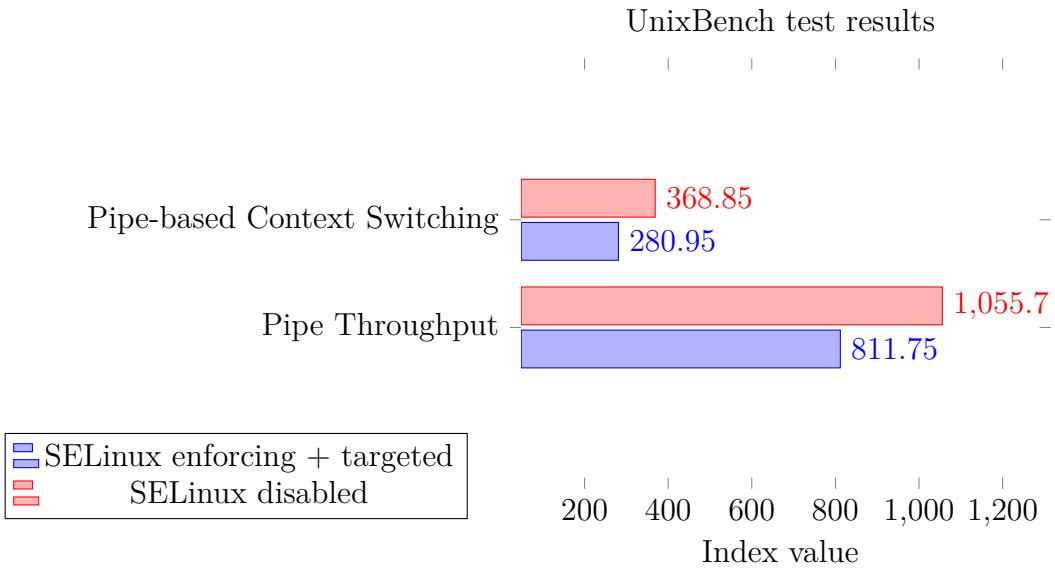

Fig. 8.7: Performance of writing and reading from the pipeline.

Twenty tests of pipe throughput and twenty tests of pipe-based context switching were run on the system, with enforcing + targeted SELinux, and then on the system with disabled SELinux. The pipe throughput average index increased, with disabled SELinux, by 23.11 % and pipe-based context switching increased by 23.83 %.

In this chapter deployed configuration usability and SELinux impact on performance was described The decrease in performance from SELinux, in enforcing mode, was between 12 and 24 %. The next chapter will show the functionality of the deployed configuration tested against a few vulnerabilities.

9 Evaluation of security configuration

In this chapter, a few proof-of-concept vulnerabilities are exploited against the following:

- SELinux in permissive mode,
- SELinux with the default configuration, shipped with Red Hat Enterprise Linux,
- deployed configuration in Ansible.

9.1 Privilege escalation with ptrace_traceme (CVE-2019-13272)

In the following listing CVE - 2019-13272[1] is demonstrated, which allows privilege escalation to root for any local user with different SELinux configurations. The description of this vulnerability on NIST website is: *"In the Linux kernel before 5.1.17, ptrace_link in kernel/ptrace.c mishandles the recording of the credentials of a process that wants to create a ptrace relationship, which allows local users to obtain root access by leveraging certain scenarios with a parent-child process relationship, where a parent drops privileges and calls execve (potentially allowing control by an attacker) [71]."*

Results of SELinux in permissive mode and default SELinux configuration were equal. It was tested on Fedora 28 with Kernel 4.16, Linux user niki wasn't confined by SELinux and the SELlinux boolean, which denies ptrace, was off.

[1] proof of concept: https://github.com/bcoles/kernel-exploits/blob/master/CVE-2019-13272/poc.c

Listing 9.1: Exploit vulnerability CVE-2019-13727 with SELinux in permissive mode.

```
[niki@localhost cve-2019-13272]$ id
uid=1001(niki) gid=1001(niki) groups=1001(niki),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

[niki@localhost cve-2019-13272]$ getsebool deny_ptrace
deny_ptrace --> off

[niki@localhost cve-2019-13272]$ ./ptrace_traceme_root
Linux 4.10 < 5.1.17 PTRACE_TRACEME local root (CVE
   -2019-13272)
[.] Checking environment ...
[~] Done, looks good
[.] Searching for known helpers ...
[~] Found known helper: /usr/libexec/gsd-wacom-led-helper
[.] Using helper: /usr/libexec/gsd-wacom-led-helper
[.] Spawning suid process (/usr/bin/pkexec) ...
[.] Tracing midpid ...
[~] Attached to midpid

[root@localhost cve-2019-13272]# id
uid=0(root) gid=0(root) groups=0(root),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

In the example, after executing ./ptrace_traceme_root, user niki obtains root access.

The next example shows executed code with the enabled SELinux boolean deny_ptrace, which is part of the SELinux configuration in Ansible.

Next example shows executed code with enabled SELinux boolean deny_ptrace which is part of SELinux configuration in Ansible.

Listing 9.2: Exploit vulnerability CVE-2019-13727 with enabled SELinux boolean deny_ptrace.

```
[niki@localhost cve-2019-13272]$ id
uid=1001(niki) gid=1001(niki) groups=1001(niki),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

[niki@localhost cve-2019-13272]$ getsebool deny_ptrace
deny_ptrace --> on

[niki@localhost cve-2019-13272]$ ./ptrace_traceme_root
Linux 4.10 < 5.1.17 PTRACE_TRACEME local root (CVE
   -2019-13272)
[.] Checking environment ...
[!] Warning: SELinux deny_ptrace is enabled
[~] Done, looks good
[.] Searching for known helpers ...
[~] Found known helper: /usr/libexec/gsd-wacom-led-helper
[.] Using helper: /usr/libexec/gsd-wacom-led-helper
[.] Spawning suid process (/usr/bin/pkexec) ...
[-] Error: ptrace(PTRACE_TRACEME, 0, NULL, NULL)
[.] Tracing midpid ...
[-] Error: ptrace(PTRACE_ATTACH, midpid, 0, NULL)

[niki@localhost cve-2019-13272]$ id
uid=1001(niki) gid=1001(niki) groups=1001(niki),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c10
```

When the boolean deny_ptrace turned on, users cannot gain root access.

In the next case the boolean deny_ptrace is disabled, but the Linux user niki is confined to SELinux user staff_u.

Listing 9.3: Exploit vulnerability CVE-2019-13727 with confined users.

```
[niki@localhost cve-2019-13272]$ id
uid=1001(niki) gid=1001(niki) groups=1001(niki),10(wheel)
context=staff_u:staff_r:staff_t:s0

[niki@localhost cve-2019-13272]$ getsebool deny_ptrace
deny_ptrace --> off

[niki@localhost cve-2019-13272]$ ./ptrace_traceme_root
Linux 4.10 < 5.1.17 PTRACE_TRACEME local root (CVE
   -2019-13272)
[.] Checking environment ...
[~] Done, looks good
[.] Searching for known helpers ...
[~] Found known helper: /usr/libexec/gsd-wacom-led-helper
[.] Using helper: /usr/libexec/gsd-wacom-led-helper
[.] Spawning suid process (/usr/bin/pkexec) ...
Error becoming real+effective uid 1001 and gid 1001:
   Operation not permitted

[niki@localhost cve-2019-13272]$ id
uid=1001(niki) gid=1001(niki) groups=1001(niki),10(wheel)
context=staff_u:staff_r:staff_t:s
```

As it can be seen, Linux users mapped into SELinux users: guest_u, staff_u, user_u, xguest_u, cannot bypass the file read, write and execute permissions check, because they are not allowed to do that by SELinux policy.

This vulnerability has an assigned category of weaknesses CWE-264: Permissions, Privileges, and Access Controls, which can lead to many technical impacts.

9.2 Privilege escalation with Dirty COW (CVE-2016-5195)

The is a privilege escalation vulnerability in the Linux Kernel, aka "Dirty COW". This is the official Red Hat description: *"A race condition was found in the way the Linux kernel's memory subsystem handled the copy-on-write (COW) breakage of private read-only memory mappings. An unprivileged local user could use this flaw to gain write access to otherwise read-only memory mappings and thus increase their privileges on the system. This could be abused by an attacker to modify existing setuid files with instructions to elevate privileges. An exploit using this technique has been found in the wild. This flaw affects most modern Linux distributions [72]."*

This vulnerability has two variants of attacks:

- write to /proc/self/mem,
- use ptrace.

For this vulnerability, there are many proofs of concept[2], using both variants of attacks. Proofs of concept of the CVE were tested on Fedora 25 with vulnerable Kernel version 4.8.0-0.

In the following example, the user is unconfined and the boolean deny_ptrace is disabled. The exploit overwrites the root account with the newly created user "attacker" with root privileges[3].

[2]proof of concepts: https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs
[3]https://github.com/FireFart/dirtycow/blob/master/dirty.c

Listing 9.4: Exploit vulnerability CVE-2019-13727 with enabled SELinux boolean deny_ptrace.

```
[niki@localhost cve-2016-5195]$ id
uid=1001(niki) gid=1001(niki) groups=1001(niki),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

[niki@localhost dirtycow]$ getsebool deny_ptrace
deny_ptrace --> off

[niki@localhost dirtycow]$ ./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line
attacker:fiVY97McoAiQ.:0:0:pwned:/root:/bin/bash

mmap: 7f66b52c9000
ptrace 0
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'attacker' and the password
   ''.

DON'T FORGET TO RESTORE!  mv /tmp/passwd.bak /etc/passwd


[niki@localhost dirtycow]$ su attacker
su attacker
Password:

[attacker@localhost dirtycow]# id
uid=0(attacker) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c10
```

Again, in this vulnerability, results of SELinux in permissive mode and default SELinux configuration were equal. But variants of the attacks, which use process tracing, were successfully blocked by the SELinux boolean deny_ptrace, when using deployed SELinux configuration.

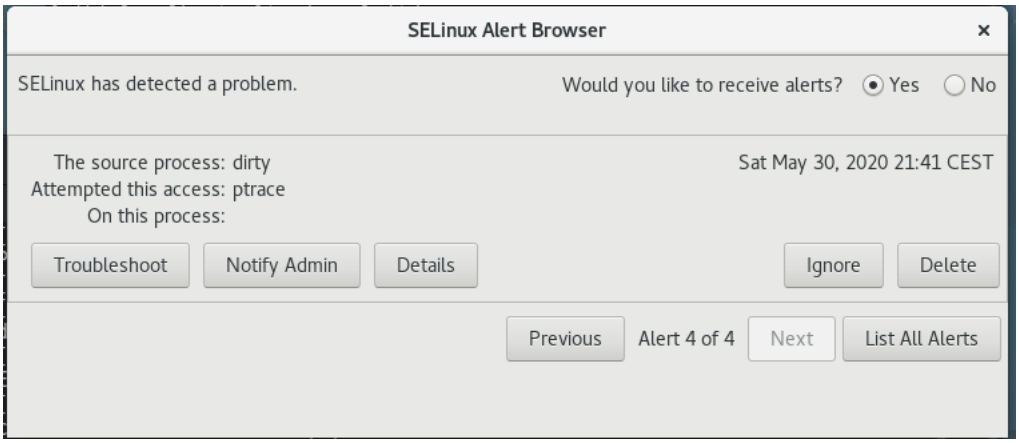

Fig. 9.1: SELinux alert: Dirty Cow.

9.3 Apache web server directory traversal

In this section a vulnerability to path traversal on an Apache webserver is shown. Through this attack, it is possible to access files and directories stored on the file system. *"By using special elements such as ".." and "/" separators, attackers can escape outside of the restricted location to access files or directories that are elsewhere on the system [73]."*

In the following examples, Apache web servers run on Fedora 30. Sequences of dots and slashes, encoded with %2F, traverse the directory and then access /home-/niki/secret.txt.

Listing 9.5: Requested URL path.

```
http://192.168.0.53/?page=..%2F..%2F..%2F..%2F..%2F..%2F..%2
    Fhome%2Fniki%2Fsecret.txt
```

Secret.txt file has permissions:

Listing 9.6: Permissions of secret.txt file.

```
-rw-r--r--. 1 root root unconfined_u:object_r:home_root_t:s0
       secret.txt
```

Which means that the owner and group is root. Root can read and write, while group and others can only read. The SELinux context is home_root_t, which is used for files in the home directory.

In the first example, SELinux is in permissive mode and with this vulnerability, it is possible to see the content of the file secret.txt.

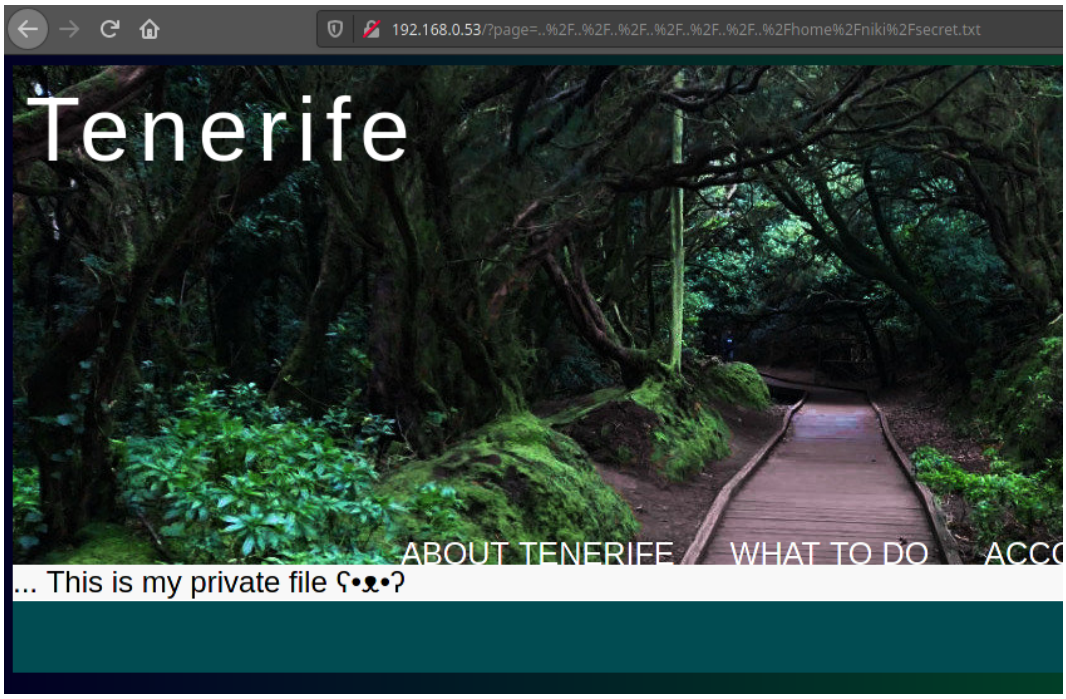

Fig. 9.2: Directory traversal with SELinux in permissive mode.

In this example, SELinux with default configuration and file secret.txt cannot be accessed by directory traversal.

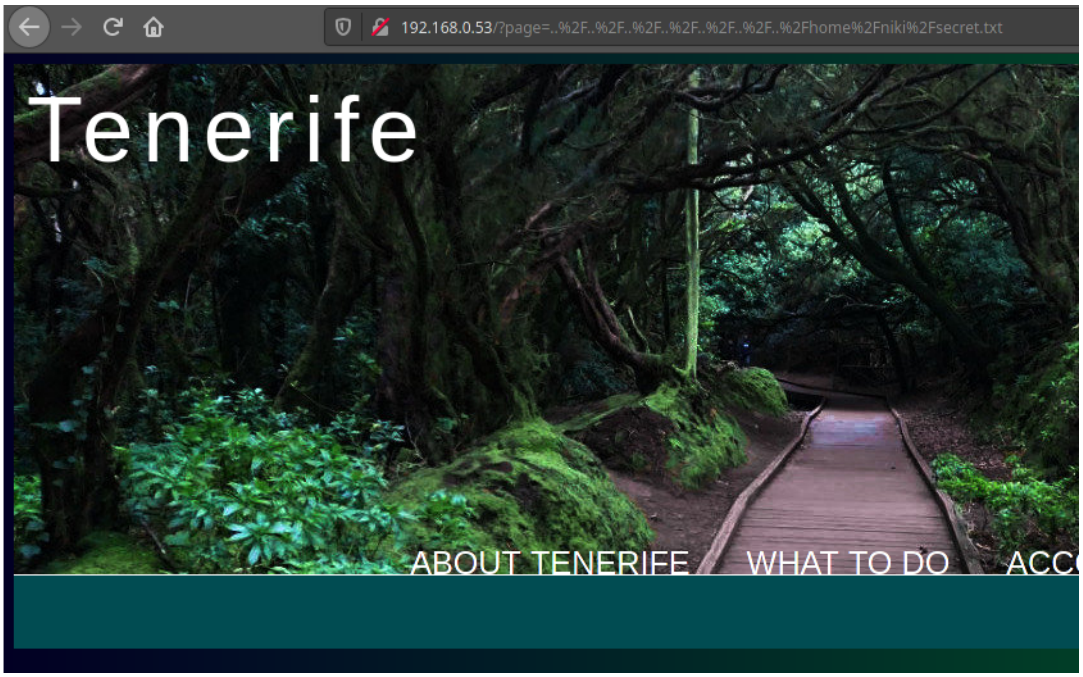

Fig. 9.3: Directory traversal with enforcing SELinux.

On the server side, a few SELinux denials have appeared, which alerts the user that the Apache process wants to open, get attributes and read the /home/niki/secret.txt file.

Fig. 9.4: SELinux alert: Directory traversal.

This is because the Apache process does not have defined access to the home directory in the SELinux policy. But, for example, an Apache process, with context httpd_t, is allowed to open or read files with context passwd_file_t defined for /etc/passwd file. So this attack is not completely blocked, but can be mitigated with enabled SELinux.

9.4 NULL pointer dereferences (CVE-2019-9213)

The default SELinux configuration was also successful against exploitation[4] of kernel NULL pointer dereferences on non-SMAP platforms. *"The vulnerability exists in the expand_downwards function, as defined in the mm/mmap.c source code of the Linux Kernel, which does not provide adequate checking for the mmap minimum address [74]."* The provided proof of concept shows the vulnerability of exploitation of kernel NULL pointer dereferences. In the first case, SELinux is in permissive mode and the vulnerability can be successfully exploited:

[4]proof of concept: https://bugs.chromium.org/p/project-zero/issues/detail?id=1792

Listing 9.7: Requested URL path.

```
[nik@localhost exploit]$ getenforce
Permissive
[nik@localhost exploit]$ ./null_map
00000000-00011000 rw-p 00000000 00:00 0
data at NULL: 0x706f2064696c6156
```

In the next case, SELinux is enforcing, with the default configuration:

Listing 9.8: Requested URL path.

```
[nik@localhost exploit]$ getenforce
Enforcing
[nik@localhost exploit]$ ./null_map
00010000-00011000 rw-p 00000000 00:00 0
Segmentation fault (core dumped)
```

This attack vector fails because SELinux has memory protection *"security check on mmap operations to see if the user is attempting to mmap to low area of the address space [18]."* This can be seen in this SELinux alert, where an attempt to access mmap_zero is logged:

Fig. 9.5: SELinux alert: NULL pointer dereferences.

No extra configuration of SELinux is needed against this vulnerability.

9.5 Code execution with PolicyKit (CVE-2018-19788)

Lukas Vrabec on his blog[5] ddemonstrated the vulnerability *" based on a bug in PolicyKit, which allows users with UID greater than INT_MAX to successfully execute any systemctl command [75]."*

This attack vector can be blocked with confined users, because SELinux users user_u, guest_u and xguest_u, cannot execute systemctl commands. Confining users is also part of the deployed Ansible configuration.

9.6 Container escape with RunC (CVE-2019-5736)

Another vulnerability described on Vrabec's blog was in runc, which *"allows malicious process in container to escape from container namespace and execute arbitrary code on host system with root privileges [76]."* Vrabec presented how default SELinux configuration blocked this attack. Enabling SELinux and setting it to "enforced" mode is part of the Ansible configuration [76].

Enabling SELinux and setting it to "enforced" mode is part of Ansible configuration.

[5]https://lukas-vrabec.com/index.php/2018/12/09/polkit-cve-2018-19788-vs-selinux/

The table displays impacts of vulnerabilities with different SELinux states and configurations.

Tab. 9.1: Impacts of vulnerabilities with different SELinux configurations.

Vulnerability description	SELinux state	SELinux configration	Impact
Privilege escalation with ptrace_traceme (CVE-2019-13272)	permisive enforcing enforcing enforcing	- default deployed deployed	privilege escalation privilege escalation blocked by deny_ptrace blocked with confined users
Privilege escalation with Dirty COW (CVE-2016-5195)	permisive enforcing enforcing	- default deployed	privilege escalation privilege escalation blocked by deny_ptrace
Apache web server directory traversal	permisive enforcing	- default	directory traversal mitigated, restricted by an Apache domain
NULL pointer dereferences (CVE-2019-9213)	permisive enforcing	- default	NULL pointer dereferences blocked, memory protection
Code execution with PolicyKit (CVE-2018-19788)	permisive enforcing enforcing	- default deployed	code execution code execution blocked with confined users
Container escape with RunC (CVE-2019-5736)	permisive enforcing	- default	container escape blocked, restricted by a container domain

This chapter provides a few examples of exploited proofs-of-concept and shows the importance of SELinux technology and its successful mitigation or blocking of exploited vulnerabilities.

Conclusion

Over the past five years, more than 8000 vulnerabilities were found, with complete information including weaknesses and technical impacts. Based on the technical impact an SELinux configuration was designed, which aims at memory protection and mitigation of privilege escalation, remote code execution, data leakage, and restriction of processes that run in unconfined and permissive domains.

The deployed configuration was done in the Ansible automation tool, which allows users to configure multiple hosts at once. SELinux is configured by managing SELinux booleans, SELinux users, and SELinux modules. Deployment of SELinux configuration prevents processes from sharing files and connecting to ports, the access of memory, and confines Linux users to SELinux users. Also in this configuration an unconfined module, which allows some processes or users to be unconfined and be minimally restricted, was disabled and placed in permissive domains.

Moreover, configurations were designed that allow users to revert SELinux deployed configurations and to lockdown the SELinux system to prevent any modifications.

Then deployed configuration functionality and impacts of the usability of the system was tested, along with how to manage them. During testing, a bug was found in the SELinux policy for Flatpak service, which is shipped by Flatpak developers. This bug was fixed and a pull request to the Flatpak GitHub repository was made. Also there was found and reported a possible security issue in the SELinux policy that allows users to bypass some restrictions. After that was the performance of a system with "enforcing + targeted SELinux" was compared to a system with disabled SELinux. Used tools were Systemd-analyze and UnixBench. The results of Systemd-analyze show that disabling SELinux increased boot-up time performance by 14 %. The results of various UnixBench tests determined that disabling SELinux increased performance from 12 to 24 %, depending on the test. The least performance impact was in system call tests, and the worst were in pipe-based context switching tests. In the last chapter, the functionality of the deployed configuration was tested against vulnerabilities. The biggest challenge was to find vulnerable versions of systems or processes and working exploits. Four vulnerabilities and weaknesses were tested against configurations targeted on privilege escalation, path traversal, and memory protection and were successfully mitigated or blocked by SELinux, with default or with deployed configurations. Moreover, in this chapter parts of SELinux configuration are described that were responsible for mitigating or blocking certain attacks. All described SELinux modifications that prevent exploiting vulnerabilities are part of the deployed SELinux configuration in Ansible.

The goal of the Master's Thesis was successfully reached. Vulnerabilities from

the previous 5 years were covered. Based on this, the SELinux configuration was designed and deployed in the Ansible automation tool. Moreover, some bugs and issues were found and fixed or reported. Even though the SELinux technology has impact on performance, the last chapter has proved the importance of SELinux technology and its successful mitigation or blocking of exploited vulnerabilities.

Bibliography

[1] ČERMÁK, Miroslav. Slabina vs. zranitelnost a jaký je mezi nimi vztah. *Clever and smart* [online]. 12. 07. 2018 [cit. 2019-12-09]. Available at: https://www.cleverandsmart.cz/slabina-vs-zranitelnost-a-jaky-je-mezi-nimi-vztah/

[2] About CWE: Category. *Common Weakness Enumeration* [online]. [cit. 2019-12-09]. Available at: https://cwe.mitre.org/documents/glossary/index.html#Category

[3] Enumeration of Technical Impacts. *Common Weakness Enumeration* [online]. [cit. 2019-12-09]. Available at: https://cwe.mitre.org/cwraf/enum_of_ti.html

[4] About CVE: Why CVE. *Common Vulnerabilities and Exposures* [online]. [cit. 2019-12-09]. Available at: https://cve.mitre.org/about/index.html#why_cve

[5] *Core Security* BUELL, Nate. A World of Vulnerabilities - Blog Post from InfoSec Institute [online]. [cit. 2019-12-09]. Available at: https://www.coresecurity.com/blog/a-world-of-vulnerabilities-guest-blog-post-from-infosec-institute

[6] Frequently Asked Questions. *Common Vulnerabilities and Exposures* [online]. [cit. 2019-12-09]. Available at: https://cve.mitre.org/about/faqs.html#what_types_of_products_use_cve

[7] Vulnerabilities. *National Vulnerability Database* [online]. [cit. 2019-12-09]. Available at: https://nvd.nist.gov/vuln/

[8] Security Data: CVE to CWE mapping. *Red Hat* [online]. [cit. 2019-12-20]. Available at: https://www.redhat.com/security/data/metrics/

[9] KIZZA, Joseph Migga. *Guide to Computer Network Security* [online]. [cit. 2019-12-09]. Available at: https://books.google.cz/books?id=xVqDgAAQBAJ&lpg=PA188&dq=%22%20initiates%20an%20access%20request%22&pg=PA188#v=onepage&q=%22%20initiates%20an%20access%20request%22&f=false

[10] DEPARTMENT OF DEFENSE COMPUTER SECURITY CENTER. *DoD Trusted Computer System Evaluation Criteria: DoD 5200.28-STD "Orange Book"* [online]. , 116 [cit. 2019-12-09]. Available at: https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/dod85.pdf

[11] THE LINUX FOUNDATION. Overview of Linux Kernel Security Features. *LINUX.COM* [online]. 11.7.2013 [cit. 2019-12-09]. Available at: https://www.linux.com/tutorials/overview-linux-kernel-security-features/

[12] Oracle Corporation *Special File Permissions* [online]. [cit. 2019-12-09]. Available at: https://docs.oracle.com/cd/E19683-01/816-4883/secfile-69/index.html

[13] The SCO Group, Inc. UnixWare 7 Documentation. *Discretionary access control DAC: permission bits* [online]. [cit. 2019-12-09]. Available at: http://uw714doc.xinuos.com/en/SEC_file/_Discretionary_Access_Control_DAC_perms.html

[14] TRESYS TECHNOLOGY, LLC. SELinux Policy Concepts and Overview: Security Policy Development Primer for Security Enhanced Linux. *Electrical Engineering and Computer Science* [online]. [cit. 2019-12-20]. Available at: http://www.cse.psu.edu/ trj1/cse543-f07/slides/03-PolicyConcepts.pdf

[15] SMALLEY, Stephen, Timothy FRASER a Chris VANCE. Linux Security Modules: General Security Hooks for Linux. *Hep* [online]. [cit. 2019-12-09]. Available at: http://www.hep.by/gnu/kernel/lsm/

[16] syscalls - Linux man page. *man7* [online]. [cit. 2019-12-21]. Available at http://man7.org/linux/man-pages/man2/syscalls.2.htm

[17] USENIX ASSOCIATION. Proceedings of the11th USENIX Security Symposium: Linux Security Modules:General Security Support for the Linux Kernel. *USENIX* [online]. San Francisco, California, USA: The USENIX Association, 2002, , 16 [cit. 2019-12-20]. Available at: https://www.usenix.org/legacy/event/sec02/full_papers/wright/wright.pdf

[18] HAINES, Richard. *The SELinux Notebook: (4th Edition)* [online]. 2014 [cit. 2019-12-21]. Available at: http://freecomputerbooks.com/books/The_SELinux_Notebook-4th_Edition.pdf

[19] Introduction to SELinux. *Red Hat* [online]. [cit. 2019-12-20]. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/ch-selinux

[20] Security Context *SELinux Project Wiki* [online]. [cit. 2019-12-20]. Available at: https://selinuxproject.org/page/NB_SC

[21] SELinux Contexts. *Red Hat* [online]. [cit. 2019-12-20]. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/chap-security-enhanced_linux-selinux_contexts

[22] SELinux User's and Administrator's Guide *Red Hat* [online]. [cit. 2019-12-20]. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html-single/selinux_users_and_administrators_guide/index

[23] Red Hat Enterprise Linux 4: Red Hat SELinux Guide: SELinux Users and Roles. *Red Hat* [online]. [cit. 2019-12-20]. Available at: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/SELinux_Guide/rhlcommon-section-0047.html

[24] WALSH, Dan. Understanding SELinux Roles. *Dan Walsh's Blog* [online]. 2016 [cit. 2019-12-20]. Available at: https://danwalsh.livejournal.com/75683.html

[25] SELinux/Type enforcement. *Gentoo linux* [online]. [cit. 2019-12-20]. Available at: https://wiki.gentoo.org/wiki/SELinux/Type_enforcement

[26] VRABEC, Lukas. Proactive Security in Linux. *Univerzita Karlova* [online]. [cit. 2019-12-20]. Available at: https://d3s.mff.cuni.cz/files/teaching/nswi161/slides/03_selinux.pdf

[27] DOMANTAS, G. What is Apache? An In-Depth Overview of Apache Web Server. *Hostinger: Tutorials* [online]. [cit. 2019-12-20]. Available at: https://www.hostinger.com/tutorials/what-is-apache

[28] Chapter 49.4. Multi-Category Security (MCS) *Red Hat* [online]. [cit. 2019-12-09]. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/sec-mcs-ov#sec-mcs-intro

[29] Security Enhanced Linux Reference Policy *Tresys Technology* [online]. [cit. 2019-12-20]. Available at: http://oss.tresys.com/docs/refpolicy/api/services_apache.html

[30] SELinux/Tutorials/Creating your own policy module file. *Gentoo linux* [online]. [cit. 2019-12-20]. Available at: https://wiki.gentoo.org/wiki/SELinux/Tutorials/Creating_your_own_policy_module_file

[31] SELinux/Policy *Gentoo linux* [online]. [cit. 2019-12-20]. Available at: https://wiki.gentoo.org/wiki/SELinux/Policy

[32] Chapter 8. Troubleshooting *CertDepot* [online]. [cit. 2019-12-09]. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/chap-security-enhanced_linux-troubleshooting

[33] Targeted Policy. *Red Hat* [online]. [cit. 2019-12-20]. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/chap-security-enhanced_linux-targeted_policy

[34] Fedora-s Mission and Foundations. *Fedora Project: fedora DOCS* [online]. [cit. 2019-12-20]. Available at: https://docs.fedoraproject.org/en-US/project/

[35] Red Hat Enterprise Linux. *Red Hat* [online]. [cit. 2019-12-20]. Available at: https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux

[36] *The CentOS Project* [online]. [cit. 2019-12-20]. Available at: https://www.centos.org/

[37] MILLER, Matthew. Fedora and CentOS Stream. *fedora MAGAZINE* [online]. [cit. 2019-12-20]. Available at: https://fedoramagazine.org/fedora-and-centos-stream/

[38] Staying close to upstream projects. *Fedora Project Wiki* [online]. [cit. 2019-12-20]. Available at: https://fedoraproject.org/wiki/Staying_close_to_upstream_projects

[39] Fedora Core 2 Release Notes. *Distrowatch* [online]. [cit. 2019-12-20]. Available at: https://distrowatch.com/external/Fedora2-Release-Notes.html

[40] RED HAT. *Red Hat Enterprise Linux 4: Red Hat SELinux Guide* [online]. 2005, , 130 [cit. 2019-12-20]. Available at: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/pdf/SELinux_Guide/Red_Hat_Enterprise_Linux-4-SELinux_Guide-en-US.pdf

[41] SELinux. *CentOS* [online]. [cit. 2019-12-20]. Available at: https://wiki.centos.org/HowTos/SELinux

[42] CWE-20: Improper Input Validation *Common Weakness Enumeration* [online]. 2019 [cit. 2019-12-20]. Available at: https://cwe.mitre.org/data/definitions/20.html

[43] WALSH, Dan. SELinux Reveals Bugs in other code.: 2006. *Dan Walsh's Blog* [online]. [cit. 2019-12-21]. Available at: https://danwalsh.livejournal.com/6117.html

[44] selinux - Linux man page *die* [online]. [cit. 2019-12-21]. Available at https://linux.die.net/man/8/selinux

[45] SELinux Booleans *CentOS* [online]. [cit. 2019-12-20]. Available at: https://wiki.centos.org/TipsAndTricks/SelinuxBooleans

[46] kernel_selinux (8) - Linux Man Pages

SysTutorials [online]. [cit. 2019-12-21]. Available at https://www.systutorials.com/docs/linux/man/8-kernel_selinux/

[47] ptrace - Linux man page. *die* [online]. [cit. 2019-12-21]. Available at http://man7.org/linux/man-pages/man2/ptrace.2.html

[48] WALSH, Dan. Fedora 17 New SELinux Feature part I - deny_ptrace.: 2013. *Dan Walsh's Blog* [online]. [cit. 2019-12-21]. https://danwalsh.livejournal.com/49336.html

[49] WALSH, Dan. Confining Samba with SELinux. *Dan Walsh's Blog* [online]. 2007 [cit. 2019-12-20]. Available at: https://danwalsh.livejournal.com/14195.html

[50] *Samba: About Samba* [online]. [cit. 2019-12-20]. Available at: https://www.samba.org

[51] samba_selinux(8) - Linux man page *die* [online]. [cit. 2019-12-21]. Available at https://linux.die.net/man/8/samba_selinux

[52] nfs_selinux(8) - Linux man page *die* [online]. [cit. 2019-12-21]. Available at https://linux.die.net/man/8/nfs_selinux

[53] GlusterFS Documentation. *GLUSTER docs* [online]. [cit. 2019-12-21]. Available at https://docs.gluster.org/en/latest/

[54] glusterd_selinux (8) - Linux Man Pages *SysTutorials* [online]. [cit. 2019-12-21]. Available at https://www.systutorials.com/docs/linux/man/8-glusterd_selinux/

[55] mysqld_selinux - SELinux Policy mysqld (8) *LinuxReviews* [online]. [cit. 2019-12-21]. Available at https://man.linuxreviews.org/man8/mysqld_selinux.8.html

[56] 4.2. Unconfined Processes. *Red Hat* [online]. [cit. 2019-12-21]. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security-enhanced_linux/sect-security-enhanced_linux-targeted_policy-unconfined_processes

[57] PermissiveDomainRecipe *SELinux Project Wiki* [online]. [cit. 2019-12-20]. Available at: https://selinuxproject.org/page/PermissiveDomainRecipe

[58] GRIFT, Dominick. SELinux Lockdown Part Nine: Booleans *SELinux Mandatory Access Control* [online]. [cit. 2019-12-20]. Available at: http://selinux-mac.blogspot.com/2009/06/selinux-lockdown-part-nine-booleans.html

[59] unconfined.if. *Github: fedora-selinux/selinux-policy* [online]. 2019 [cit. 2019-12-20]. Available at: https://github.com/fedora-selinux/selinux-policy/blob/rawhide/policy/modules/system/unconfined.if

[60] PolicyStatements*SELinux Project Wiki* [online]. [cit. 2019-12-20]. Available at: https://selinuxproject.org/page/PolicyStatements

[61] Basic Concepts. *Ansible Project: Documentation* [online]. [cit. 2019-12-20]. Available at: https://docs.ansible.com/ansible/latest/network/getting_started/basic_concepts.html

[62] Roles. *Ansible Project: Documentation* [online]. [cit. 2019-12-20]. Available at: https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

[63] Red Hat Enterprise Linux System Roles Powered by Ansible. *Red Hat* [online]. [cit. 2019-12-21]. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/7.6_release_notes/new_features_red-_hat_enterprise_linux_system_roles_powered_by_ansible

[64] SELinux. *Github* [online]. 2019 [cit. 2019-12-20]. Available at: https://github.com/linux-system-roles/selinux/blob/master/README.md

[65] How to build your inventory. *Ansible Project: Documentation* [online]. [cit. 2019-12-20]. Available at: https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

[66] WALSH, Dan. Difference between a Confined User (staff_u) and a Confined Administrator. *Dan Walsh's Blog* [online]. 2007 [cit. 2019-12-20]. Available at: https://danwalsh.livejournal.com/66587.html

[67] systemd-analyze — Analyze and debug system manager *freedesktop.org* [online]. 2019 [cit. 2019-12-21]. Available at: https://www.freedesktop.org/software/systemd/man/systemd-analyze.html

[68] kdlucas/byte-unixbench. *Github* [online]. 2019 [cit. 2019-12-20]. Available at: https://github.com/kdlucas/byte-unixbench/blob/master/UnixBench/USAGE

[69] QIAN Chao. *UnixBench: A Detailed Implementation* [online]. 2019 [cit. 2019-12-20]. Available at: https://medium.com/@Alibaba_Cloud/unixbench-a-detailed-implementation-41d17f6352a5

[70] FAHEEM P. *UnixBench: Find your system performance.* [online]. 2019 [cit. 2019-12-20]. Available at: https://www.supportsages.com/unixbench-find-your-system-performance/

[71] CVE-2019-13272. *National Vulnerability Database* [online]. 2019 [cit. 2019-12-21]. Available at: https://nvd.nist.gov/vuln/detail/CVE-2019-13272

[72] Kernel Local Privilege Escalation "Dirty COW" - CVE-2016-5195 *Red Hat* [online]. 2019 [cit. 2019-12-21]. Available at: https://access.redhat.com/security/vulnerabilities/DirtyCow

[73] CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') *Common Weakness Enumeration* [online]. 2019 [cit. 2019-12-20]. Available at: https://cwe.mitre.org/data/definitions/22.html

[74] Linux Kernel expand_downwards Function NULL Pointer Dereference Vulnerability *Niss.dk* [online]. 2019 [cit. 2019-12-21]. Available at: https://niss.dk/2019/03/07/linux-kernel-expand_downwards-function-null-pointer-dereference-vulnerability/

[75] Polkit CVE-2018-19788 vs. SELinux *Lukas Vrabec* [online]. 2019 [cit. 2019-12-21]. Available at: https://lukas-vrabec.com/index.php/2018/12/09/polkit-cve-2018-19788-vs-selinux

[76] CVE-2019-5736 runc escape vs. SELinux *Lukas Vrabec* [online]. 2019 [cit. 2019-12-21]. Available at: https://lukas-vrabec.com/index.php/2019/04/05/cve-2019-5736-runc-escape-vs-selinux

List of symbols, physical constants and abbreviations

CVE	Common Vulnerabilities and Exposures
CWE	Common weakness enumeration
DAC	Discretionary Access Control
LSM	Linux Security Modules
MAC	Mandatory Access Control
MCS	Multi-Category Security
MLS	Multi-Level Security
NFS	Network File System
NSA	National Security Agency
NVD	National Vulnerability Database
RBAC	Role-Based Access Contro
RHEL	Red Hat Enterprise Linux
SELinux	Security-Enhanced Linux
TE	Type enforcement

List of appendices

A Ansible configuration file deny_execmem

Listing A.1: Ansible configuration file deny_execmem.

```
#Turn off all booleans which allow domain execmem/execstack
deny_execmem:
# Determine whether boinc can execmem/execstack.
      - { name: 'boinc_execmem', state: 'off', persistent: '
        yes' }

# Allow cluster administrative cluster domains memcheck-amd64
   - to use executable memory
      - { name: 'cluster_use_execmem', state: 'off',
        persistent: 'yes' }

# Allow cups execmem/execstack
      - { name: 'cups_execmem', state: 'off', persistent: 'yes
        ' }

# Deny user domains applications to map a memory region as
   both executable and writable
      - { name: 'deny_execmem', state: 'on', persistent: 'yes'
         }

# Allow glance domain to use executable memory and executable
    stack
      - { name: 'glance_use_execmem', state: 'off', persistent
        : 'yes' }

# Allow glusterd_t domain to use executable memory
      - { name: 'gluster_use_execmem', state: 'off',
        persistent: 'yes' }

# Allow httpd scripts and modules execmem/execstack
      - { name: 'httpd_execmem', state: 'off', persistent: '
        yes' }

# Determine whether mplayer can make its stack executable
      - { name: 'mplayer_execstack', state: 'off', persistent
        : 'yes' }
```

```
# Allow unconfined executables to make their heap memory executable. Doing this is a really bad idea. Probably indicates a badly coded executable, but could indicate an attack. This executable should be reported in bugzilla
    - { name: 'selinuxuser_execheap', state: 'off', persistent: 'yes' }

# Allow all unconfined executables to use libraries requiring text relocation that are not labeled textrel_shlib_t
    - { name: 'selinuxuser_execmod', state: 'off', persistent: 'yes' }

# Allow unconfined executables to make their stack executable. This should never, ever be necessary. Probably indicates a badly coded executable, but could indicate an attack. This executable should be reported in bugzilla
    - { name: 'selinuxuser_execstack', state: 'off', persistent: 'yes' }

# Allow tomcat to use executable memory and executable stack
    - { name: 'tomcat_use_execmem', state: 'off', persistent: 'yes' }

# Allow confined virtual guests to use executable memory and executable stack
    - { name: 'virt_use_execmem', state: 'off', persistent: 'yes' }

# Allows XServer to execute writable memory
    - { name: 'xserver_execmem', state: 'off', persistent: 'yes' }
```

B Ansible configuration file deny_export

Listing B.1: Ansible configuration file deny_export.

```
deny_export:
### Gluster ###

# Allow glusterfsd to modify public files used for public
   file transfer services. Files/Directories must be labeled
   public_content_rw_t.
      - { name: 'gluster_anon_write', state: 'off', persistent
        : 'yes' }

# Allow glusterfsd to share any file/directory read/write
      - { name: 'gluster_export_all_rw', state: 'off',
        persistent: 'yes' }

# Allow glusterfsd to share any file/directory read only.
      - { name: 'gluster_export_all_ro', state: 'off',
        persistent: 'yes' }

### NFS ###

# Allow nfs servers to modify public files used for public
   file transfer services. Files/Directories must be labeled
   public_content_rw_t
      - { name: 'nfsd_anon_write', state: 'off', persistent: '
        yes' }

# Allow nfs to export all read/write
      - { name: 'nfs_export_all_rw', state: 'off', persistent
        : 'yes' }

# Allow nfs to export all read only.
      - { name: 'nfs_export_all_ro', state: 'off', persistent
        : 'yes' }

### Samba ###

# Allow samba to modify public files used for public file
   transfer services.  Files/Directories must be labeled
   public_content_rw_t.
```

```
      - { name: 'smbd_anon_write', state: 'off', persistent: '
        yes' }

# Allow samba to share users home directories.
      - { name: 'samba_enable_home_dirs', state: 'off',
        persistent: 'yes' }

# Allow samba to export all read/write
      - { name: 'samba_export_all_rw', state: 'off',
        persistent: 'yes' }

# Allow samba to export all read only.
      - { name: 'samba_export_all_ro', state: 'off',
        persistent: 'yes' }

# Allow samba to export ntfs/fusefs volumes.
      - { name: 'samba_share_fusefs', state: 'off', persistent
        : 'yes' }

# Allow samba to export NFS volumes.
      - { name: 'samba_share_nfs', state: 'off', persistent: '
        yes' }
```

C Ansible configuration file cant_connect

Listing C.1: Ansible configuration file cant_connect.

```
#Turn off all booleans which allow domain connect to any network port
cant_connect:

# Allow cluster administrative domains to connect to the network using TCP.
    - { name: 'cluster_can_network_connect', state: 'off', persistent: 'yes' }

# Determine whether Cobbler can connect to the network using TCP.
    - { name: 'cobbler_can_network_connect', state: 'off', persistent: 'yes' }

# Determine whether collectd can connect to the network using TCP.
    - { name: 'collectd_tcp_network_connect', state: 'off', persistent: 'yes' }

# Determine whether conman can connect to all TCP ports
    - { name: 'conman_can_network', state: 'off', persistent: 'yes' }

# Determine whether Condor can connect to the network using TCP.
    - { name: 'condor_tcp_network_connect', state: 'off', persistent: 'yes' }

# Determine whether exim can connect to databases.
    - { name: 'exim_can_connect_db', state: 'off', persistent: 'yes' }

# Determine whether fenced can connect to the TCP network.
    - { name: 'fenced_can_network_connect', state: 'off', persistent: 'yes' }

# Determine whether ftpd can connect to all unreserved ports.
```

```
      - { name: 'ftpd_connect_all_unreserved', state: 'off',
        persistent: 'yes' }

# Determine whether ftpd can bind to all unreserved ports for
    passive mode.
      - { name: 'ftpd_use_passive_mode', state: 'off',
        persistent: 'yes' }

# Determine whether ftpd can connect to databases over the
   TCP network.
      - { name: 'ftpd_connect_db', state: 'off', persistent: '
        yes' }

# Determine whether Git session daemon can bind TCP sockets
   to all unreserved ports.
      - { name: 'git_session_bind_all_unreserved_ports', state
        : 'off', persistent: 'yes' }

# Determine whether glance-api can connect to all TCP ports
      - { name: 'glance_api_can_network', state: 'off',
        persistent: 'yes' }

# Determine whether haproxy can connect to all TCP ports.
      - { name: 'haproxy_connect_any', state: 'off',
        persistent: 'yes' }

# Allow httpd to act as a FTP client connecting to the ftp
   port and ephemeral ports
      - { name: 'httpd_can_connect_ftp', state: 'off',
        persistent: 'yes' }

# Allow httpd to connect to the ldap port
      - { name: 'httpd_can_connect_ldap', state: 'off',
        persistent: 'yes' }

# Allow http daemon to connect to mythtv
      - { name: 'httpd_can_connect_mythtv', state: 'off',
        persistent: 'yes' }

# Allow http daemon to connect to zabbix
```

```
      - { name: 'httpd_can_connect_zabbix', state: 'off',
        persistent: 'yes' }

# Allow HTTPD scripts and modules to connect to the network
   using TCP.
      - { name: 'httpd_can_network_connect', state: 'off',
        persistent: 'yes' }

# Allow HTTPD scripts and modules to connect to cobbler over
   the network.
      - { name: 'httpd_can_network_connect_cobbler', state: '
        off', persistent: 'yes' }

# Allow HTTPD scripts and modules to connect to databases
   over the network.
      - { name: 'httpd_can_network_connect_db', state: 'off',
        persistent: 'yes' }

# Allow httpd to connect to memcache server
      - { name: 'httpd_can_network_memcache', state: 'off',
        persistent: 'yes' }

# Allow httpd to act as a relay
      - { name: 'httpd_can_network_relay', state: 'off',
        persistent: 'yes' }

# Allow httpd to act as a FTP server by listening on the ftp
   port.
      - { name: 'httpd_enable_ftp_server', state: 'off',
        persistent: 'yes' }

# Allow HTTPD to connect to port 80 for graceful shutdown
      - { name: 'httpd_graceful_shutdown', state: 'off',
        persistent: 'yes' }

# Allow httpd to access openstack ports
      - { name: 'httpd_use_openstack', state: 'off',
        persistent: 'yes' }

# Allow httpd to connect to sasl
```

```
      - { name: 'httpd_use_sasl', state: 'off', persistent: '
        yes' }

# Determine whether icecast can listen on and connect to any
   TCP port.
      - { name: 'icecast_use_any_tcp_ports', state: 'off',
        persistent: 'yes' }

# Determine whether irc clients can listen on and connect to
   any unreserved TCP ports.
      - { name: 'irc_use_any_tcp_ports', state: 'off',
        persistent: 'yes' }

# Allow the Irssi IRC Client to connect to any port, and to
   bind to any unreserved port.
      - { name: 'irssi_use_full_network', state: 'off',
        persistent: 'yes' }

# Determine whether keepalived can connect to all TCP ports.
      - { name: 'keepalived_connect_any', state: 'off',
        persistent: 'yes' }

# Determine whether logwatch can connect to mail over the
   network.
      - { name: 'logwatch_can_network_connect_mail', state: '
        off', persistent: 'yes' }

# Determine whether lsmd_plugin can connect to all TCP ports.
      - { name: 'lsmd_plugin_connect_any', state: 'off',
        persistent: 'yes' }

# Allow mozilla plugin domain to bind unreserved tcp/udp
   ports.
      - { name: 'mozilla_plugin_bind_unreserved_ports', state
        : 'off', persistent: 'yes' }

# Allow mozilla plugin domain to connect to the network using
    TCP.
      - { name: 'mozilla_plugin_can_network_connect', state: '
        off', persistent: 'yes' }
```

```
# Allow mysqld to connect to all ports
      - { name: 'mysql_connect_any', state: 'off', persistent
        : 'yes' }

# Allow mysqld to connect to http port
      - { name: 'mysql_connect_http', state: 'off', persistent
        : 'yes' }

# Determine whether Bind can bind tcp socket to http ports.
      - { name: 'named_tcp_bind_http_port', state: 'off',
        persistent: 'yes' }

# Determine whether neutron can connect to all TCP ports
      - { name: 'neutron_can_network', state: 'off',
        persistent: 'yes' }

# Determine whether openvpn can connect to the TCP network.
      - { name: 'openvpn_can_network_connect', state: 'off',
        persistent: 'yes' }

# Allow pcp to bind to all unreserved_ports
      - { name: 'pcp_bind_all_unreserved_ports', state: 'off',
         persistent: 'yes' }

# Allow PowerDNS to connect to databases over the network.
      - { name: 'pdns_can_network_connect_db', state: 'off',
        persistent: 'yes' }

# Allow piranha-lvs domain to connect to the network using
   TCP.
      - { name: 'piranha_lvs_can_network_connect', state: 'off
        ', persistent: 'yes' }

# Determine whether Polipo session daemon can bind tcp
   sockets to all unreserved ports.
      - { name: 'polipo_session_bind_all_unreserved_ports',
        state: 'off', persistent: 'yes' }

# Allow polipo to connect to all ports > 1023
      - { name: 'polipo_connect_all_unreserved', state: 'off',
         persistent: 'yes' }
```

```
                                                                    135
# Determine whether privoxy can connect to all tcp ports.           136
      - { name: 'privoxy_connect_any', state: 'off',                137
        persistent: 'yes' }
                                                                    138
# Permit to prosody to bind apache port. Need to be activated       139
    to use BOSH.
      - { name: 'prosody_bind_http_port', state: 'off',             140
        persistent: 'yes' }
                                                                    141
# Allow samba to act as a portmapper                                142
      - { name: 'samba_portmapper', state: 'off', persistent:       143
        'yes' }
                                                                    144
# Allow users to connect to the local mysql server                  145
      - { name: 'selinuxuser_mysql_connect_enabled', state: '       146
        off', persistent: 'yes' }
                                                                    147
# Allow users to connect to PostgreSQL                              148
      - { name: 'selinuxuser_postgresql_connect_enabled',           149
        state: 'off', persistent: 'yes' }
                                                                    150
# Allow users to run TCP servers (bind to ports and accept          151
   connection from the same domain and outside users)
   disabling this forces FTP passive mode and may change
   other protocols.
      - { name: 'selinuxuser_tcp_server', state: 'off',             152
        persistent: 'yes' }
                                                                    153
# Allow users to run UDP servers (bind to ports and accept          154
   connection from the same domain and outside users)
   disabling this may break avahi discovering services on the
    network and other udp related services.
      - { name: 'selinuxuser_udp_server', state: 'off',             155
        persistent: 'yes' }
                                                                    156
# Allow sge to connect to the network using any TCP port            157
      - { name: 'sge_domain_can_network_connect', state: 'off'      158
        , persistent: 'yes' }
                                                                    159
# Allow user spamassassin clients to use the network.               160
```

```
      - { name: 'spamassassin_can_network', state: 'off',
        persistent: 'yes' }

# Allow spamd_update to connect to all ports.
      - { name: 'spamd_update_can_network', state: 'off',
        persistent: 'yes' }

# Determine whether squid can connect to all TCP ports.
      - { name: 'squid_connect_any', state: 'off', persistent
        : 'yes' }

# Determine whether sslh can listen on any tcp port or if it
   is restricted to the standard http.
      - { name: 'sslh_can_bind_any_port', state: 'off',
        persistent: 'yes' }

# Determine whether sslh can connect to any tcp port or if it
    is restricted to the standard http, openvpn and jabber
   ports.
      - { name: 'sslh_can_connect_any_port', state: 'off',
        persistent: 'yes' }

# Determine whether swift can connect to all TCP ports
      - { name: 'swift_can_network', state: 'off', persistent
        : 'yes' }

# Allow the Telepathy connection managers to connect to any
   network port.
      - { name: 'telepathy_connect_all_ports', state: 'off',
        persistent: 'yes' }

# Allow the Telepathy connection managers to connect to any
   generic TCP port.
      - { name: 'telepathy_tcp_connect_generic_network_ports',
         state: 'off', persistent: 'yes' }

# Allow tomcat to connect to databases over the network.
      - { name: 'tomcat_can_network_connect_db', state: 'off',
         persistent: 'yes' }
```

```
# Determine whether tor can bind tcp sockets to all unreserved ports.
    - { name: 'tor_bind_all_unreserved_ports', state: 'off', persistent: 'yes' }

# Allow tor to act as a relay
    - { name: 'tor_can_network_relay', state: 'off', persistent: 'yes' }

# Determine whether varnishd can use the full TCP network.
    - { name: 'varnishd_connect_any', state: 'off', persistent: 'yes' }

# Allow confined virtual guests to use serial/parallel communication ports
    - { name: 'virt_use_execmem', state: 'off', persistent: 'yes' }

# Allows xdm_t to bind on vnc_port_t
    - { name: 'xdm_bind_vnc_tcp_port', state: 'off', persistent: 'yes' }

# Determine whether zabbix can connect to all TCP ports
    - { name: 'zabbix_can_network', state: 'off', persistent: 'yes' }
```