VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTEGRACE SELINUX AUDIT ZÁZNAMŮ DO ABRT NÁSTROJE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE LUKÁŠ VRABEC
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTEGRACE SELINUX AUDIT ZÁZNAMŮ DO ABRT NÁSTROJE

INTEGRATION OF SELINUX AUDIT LOGS INTO ABRT TOOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ VRABEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. BARABAS MAROŠ
SUPERVISOR

BRNO 2015

Abstrakt

Cílem práce je seznámit čtenáře s problematikou bezpečnosti operačních systémů Linux a bezpečnostním mechanismem řízení přístupu. Dále se práce zabývá zpracováním bezpečnostních zpráv, které jsou produkovány bezpečnostními mechanismy, do záznamových služeb a jejich zobrazením. V práci je obsažen návrh a implementace řešení, které integruje bezpečnostní zprávy do centralizovaného systému, uchovává je a nahlašuje chyby v komunitní linuxové distribuci Fedora a komerční distribuci Red Hat Enterprise Linux.

Abstract

The main aim of the thesis is to introduce the security of Linux operating systems and the access control list mechanism. The thesis focuses on processing security messages produced by the security mechanism into logging services and on displaying those messages. In addition, the thesis includes a concept solution and its implementation, which integrates security messages into a centralized system, keeps them and reports bugs in the Linux community distribution Fedora and in the Linux commercial distribution Red Hat Enterprise Linux.

Klíčová slova

Bezpečnosť, Bezpečnostné mechanizmy, Riadenie prístupu, SELinux, Bezpečnostná politika, Záznamová služba, Syslog, Systemd-journald, Auditovací systém, Bugzilla, ABRT, MAC, DAC, AVC správa

Keywords

Security, Serucity mechanizm, Access control, SELinix, Security policy, Logging service, Syslog, Systemd-journald, Audit system, Bugzilla, ABRT, MAC, DAC, AVC message

Citace

Lukáš Vrabec: Integrace SELinux audit záznamů do ABRT nástroje, bakalářská práce, Brno, FIT VUT v Brně, 2015

Integrace SELinux audit záznamů do ABRT nástroje

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Maroša Barabasa.

. . . . . .
Lukáš Vrabec
17. května 2015

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Marošovi Barabasovi za odbornou pomoc, cenné rady a čas, který mi věnoval. Také bych rád poděkoval Ing. Miroslavovi Greplovi za odborné rady a dlouhodobou spolupráci na projektě SELinux. V neposlední řadě děkuji i firmě Red Hat Inc. za poskytnutí možnosti pracovat na této práci.

Obsah

Kapitola 1

Úvod

Operačný systém Linux [20] je pomerne rozšírený a jednoduchý operačný systém. Z dôvodu častejšieho nasadzovania do prevádzky je nutné dbať aj na jeho bezpečnosť [21]. Podstatou zabezpečovania operačných systémov je snaha znížiť náchylnosť operačného systému na rôzne kybernetické útoky. Medzi najrozšírenejšie kybernetické útoky patrí útok pretečenia zásobníku (tzv. "buffer overflow") vďaka ktorému, je možné získať práva superuživateľa. Z dôvodu detekcie a prevencie pred podobnými útokmi sú vyvíjané rôzne formy bezpečnostných mechanizmov.

Prvým faktorom pri zabezpečení je analýza samotného systému a navrhnutie relevatných bezpečnostných mechanizmov ako je napr. Firewall alebo SELinux [10]. Takéto bezpečnostné mechanizmy pracujú na rôznych princípoch, napr. sledovaním sieťovej aktivity v operačnom systéme, izolácií sputených procesov[1], alebo citlivým nastavením prístupových práv. Medzi najznámejšie a najmasívnejšie bezpečnostné mechanizmy patrí Security Enhanced Linux (SELinux) systém. SELinux vymedzuje rozsah činnosti aplikáciam a službám, čím poskytuje vysoký stupeň bezpečnosti operačného systému.

Druhým dôležitým faktorom pri zabezpečovaní operačného sýstému je nájdenie správneho pomeru medzi prívetivosťou a bezpečnosťou. Uživateľ vyžaduje bezpečný systém, ktorý chráni jeho data a zároveň ho neobmedzuje pri práci. Za účeľom zlepšenia prívetivosti bol vyvinutý nástroj ABRT (*Automatic Bug Reporting Tool*). ABRT pomáha uživateľovi jednodnucho detekovať chyby a vady v systéme. Zhromažduje záznamy a informácie o chybách, ktoré môžu byť odoslané správcovi chybného balíčku na opravu.

Problematika kompromisu medzi adekvátnou prívetivosťou a zabezpečením je v dnešnej dobe veľmi populárna. Či už z hľadiska bezpečnosti, kedy počet a sofistikovanosť útokov ročne narastá, tak aj z hľadiska čo najmenšieho obmedzenia funkcionality operačného systému pri bezpečnostných mechanizmoch.

1.1 Cieľ práce

Najdôležitejším cieľom tejto práce je implementovanie neexistujúcej integrácie SELinux záznamov do jednotnej aplikácie pre zhromažďovanie systémových chýb a zlyhaní (Automatic bug reporting - ABRT) a prípadné nahlasovanie týchto problémov vývojárom komunitnej distribúcie Fedora, alebo komerčnej distribúcie Red Hat Enterprise Linux. Ďalším dôvodom je vytvorenie rozhrania, ktoré bude ukladať prehľadným spôsobom SELinux záznamy do systémových žurnálov (journald).

[1] Proces je bežiaca instancia programu vrátane stavu a hodnôt premenných

Pomocou týchto spôsobov sa očakáva zlepšenie užívateľskej prívetivosti a jednoduchšiemu spracovaniu správ bezpečnostného mechanizmu SELinux. Rovnako dôležitá je aj implementácia novej časti grafického uživateľského rozhrania v nástroji ABRT, na zobrazovanie bezpečnostných hlásení, čo bude mať za následok minimalizovanie nahlasovania falošných chýb v SELinuxe.

1.2 Štruktúra práce

V nasledujúcej časti budú stručne popísané jednotlivé kapitoly, ktorými sa táto práca zaoberá. Každá z nich sa zaoberá teoretickým alebo praktickým vysvetlením projektu integrácie.

Kapitola 2: Táto teoretická kapitola je venovaná bezpečnostným mechanizmom, ktorých výstup bude spracovávaný do jednej zo záznamových služieb.

Kapitola 3: Kapitola venujúca teoretickým základom využitia záznamových služieb ako je syslog a systemd-journald.

Kapitola 4: V kapitole je obsiahnutý rozbor problému na jednotlivé podproblémy, na ktorých následne stavia návrh implementácie. Sú zadefinované funkčné požiadavky integrácie.

Kapitola 5: Návrh implementácie podrobne popisuje navrhovanie integrácie. Súčasťou tejto kapitoly je aj grafický návrh užívateľského rozhrania.

Kapitola 6: Zhrnutie implementácie, pričom je kladený dôraz na postup implementácie a implementačné problémy. V tejto kapitole sú popísané aj použité technológie.

Kapitola 7: Záver a popis práce v posledných fázach spolu so zhodnotením celkovej práce. Ďalej je predstavená budúcnosť projektu a návrhy na zlepšenie práce.

Kapitola 2

Bezpečnostné mechanizmy

V dnešnej dobe je veľmi obtiažne vyvinúť bezchybný a dokonale bezpečný program. Programy alebo služby sú náchylné na rôzne typy útokov, ktoré môžu viesť až k úplnej kontrole celého systému. Z týchto dôvodov sú používané rôzne formy bezpečnostných mechanizmov. Jedným z týchto mechanizmov je aj **kontrola riadenia prístupu** [14].

2.1 Kontrola riadenia prístupu

Operačné systémy používajú rôzne mechanizmy na riadenie prístupu [6] ku jednotlivým objektom(súborom, adresárom, socketom, file descriptorom[1] a pod). Najznámejším mechanizmom je diskrétne riadenie prístupových práv (anglicky *DAC - Discretionary access control*), ktorý je implementovaný aj v jadre operačného systému Linux. Druhým známym riadením prístupu je povinné riadenie prístupu (anglicky *MAC - Mandatory access control*). Oba tieto prístupy budú vysvetlené v nasledujúcich kapitolách.

2.1.1 DAC (Discretionary access control)

Nepovinné riadenie prístupu [3] je druh riadenia prístupu zabezpečenia, ktoré udeľuje alebo obmedzuje prístup k objektu, typicky súborového systému, na základe identity vlastníka alebo skupiny do ktorej uživateľ žiadajúci prístup k objektu patrí. Tento druh sa nazýva nepovinný, nakoľko vlastník tohto objektu môže meniť atribúty(vlastnosti) tohto objektu.

DAC bezpečnostný mechanizmus používa tzv. ACL(*Access control list*) zoznamy. Každý súbor má pridelený jeden ACL zoznam, kde sú zadefinované pravidlá pre vlastníka súboru, skupinu uživateľov a ostatných uživateľov. Na základe ACL zoznamu je teda povolené alebo zablokované čítanie, zapisovanie alebo spúštanie konkrétneho súboru.

Tento prístup má jednu veľkú nevýhodu a to takú, že všetky spustené procesy sú spustené s právami uživateľa, ktorý tento proces vytvoril. To znamená, že daný proces má prístup k všetkým súborom, ku ktorým má prístup aj uživateľ, čo znamená že proces môže tieto súbory čítať alebo v horšom prípade modifikovať, či už úmyselne alebo neúmyselne.

Napríklad poštový klient nepotrebuje mať prístup k všetkým súborom uživateľa, ktoré obsahujú konfiguračné súbory alebo rôzne citlivé dáta, ktoré môžu byť prečítané alebo modifikované. Najhorším prípadom je spustenie procesu pod právami superuživateľa. Takýto proces má prístup ku všetkým súborom v celom operačnom systéme, čo môže viesť k fatálnym následkom v prípade napadnutia takéhoto procesu.

[1] popisovačom súboru

Rovnaké nebezpečenstvo hrozí, aj keď má proces nastavený *setuid*, alebo *setgid* bit na superužívateľa. Z týchto dôvodov je používanie len nepovinného riadenia prístupu z bezpečnostného hľadiska nedostatočné. Možným riešením je použitie implementácie povinného riadenia prístupu.

2.1.2 MAC (Mandatory access control)

Povinné riadenie prístupu [7] umožňuje úplnu kontrolu nad všetkými akciami procesov, ktoré bežia v tzv. *sandboxe*, ktorý obmedzuje prístup do iných častí systému, čo je samotný priestor *sandboxu*. SELinux používa pre termín *sandbox* názov *doména*.

Tento mechanizmus zabezpečuje, aby proces nedokázal opustiť jeho doménu, a teda aby nemal prístup k súborom, ktoré nie sú potrebné pre bezproblémový chod daného procesu. V praxi to znamená, že ak sa podarí útočníkovi prevziať kontrolu nad procesom, môže vykonávať len také operácie, ktoré sú povolené v doméne, v ktorej sa proces nachádza. To platí aj v prípade, že proces spustil superužívateľ.

Všetky domény musia mať špecifikované povolovacie pravidlá v bezpečnostných politikách, ktoré zadefinujú rozsah ackií, ktoré môže proces v doméne vykonať.

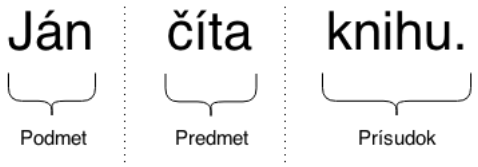
V tabuľke(2.1) sú popísané hlavné rozdiely medzi oboma typmi riadenia prístupu.

DAC	**MAC**
proces môže meniť bezpečnostný atribút	proces nemôže meniť bezpečnostný atribút
bezpečnostná politika je zakódovaná v jadre	je potrebné bezpečnostnú politiku definovať
viac úrovní oprávenia (superuživateľ, užívateľ)	jedna úroveň oprávenia (neexistuje superuživateľ)

Tabulka 2.1: Rozdiely DAC a MAC

2.2 Objekty a subjekty

Vysvetlenie nasledujúcich pojmov je možné vysvetliť na všeobecnom príklade. Význam je veľmi podobný ako v prípade gramatického členenia slov vo vete v slovenskom rovnako aj českom jazyku. Je zadaná veta: *Ján číta knihu.* Jednotlivé slová vo vete je možné rozdeliť nasledovne: Kdo číta knihu? Ján(Podmet). Čo robí Ján? Číta(Predmet). Čo Ján číta? Knihu(Prísudok).



Obrázek 2.1: členenie vety

Z predchádzajúcej vety je umožné identifikovať jednoznačnú akciu, ktorá sa deje v reálnom svete. Pomocou pojmov subjekt a objekt je možné rovnako zostaviť akciu, ktorú bude

možné jednoznačne identifikovať v počítačovom systéme. Podmet je možné nahradiť subjektom, prísudok je možné nahradiť objektom a predmet operáciu. Z týchto pojmov je možné zostaviť bezpečnostný model, ktorý je rozobraný v nasledujúcej časti.
Bezpečnostný model obsahuje tri základné elementy, s ktorými pracuje(viď. Tabuľka 2.2).

Element	**Popis**
Subjekty	Aktéri
Objekty	Súbory, sockety
Operácie	Akcie

Tabulka 2.2: Elementy bezpečnostného modelu

Subjekty sú aktéri v počítačovom systéme. Spočiatku často dochádza k mylnému pochopeniu tohto významu. Pojem subjekt neoznačuje užívateľa v operačnom systéme ale označuje bežiaci proces. Napriek tomu je možné si predstaviť proces, ktorý zastupuje užívateľa pri vykonávaní jednotlivých akcií. To znamená, že subjekty a užívatelia sú úzko prepojení aj keď procesy sú skutoční aktéri.

Dôležité je spomenúť rozdieľ medzi procesom a programom. Program je spustiteľný súbor, zatiaľ čo proces je program, ktorý bol načítaný do pamäte a bolo zahájené jeho spustenie. Napríklad, ak sú spustené dve konzolové okná v grafickom rozhraní, sú spustené dva procesy jedného programu. Narozdieľ od programu, proces poskytuje informácie o svojom stave. Tieto informácie môžu byť nasledovné: Identita vlastníka procesu, hodnoty aktívnych premenných v programe a pod.

Rovnako ako aj v gramatikách rôzných jazykov(napr. Slovenský jazyk) podmet operuje nad prísudkom. To isté platí aj v popisovanom bezpečnostnom modely, kedy subjekt(proces) rovnako pracuje nad objektom(súborom).

Bezpečnostný model definuje niekoľko desiatok bezpečnostných tried objektov, napríklad: *Adresáre, Súbory, Súborové systémy, Odkazy ,File descriptors(Popisovač súboru), Sockety (schránky)* [9]. Proces môže byť rovnako ako subjekt aj objekt. Operácie v bezpečnostnom modely definujú aký typ operácie môže subjekt vykonať nad objektom. Zoznam najčastejšie používaných operácií: *Vytvoriť, Spustiť, Čítať, Zapísať, Pridať, Premenovať, Získať atribúty, Uzamknúť.*

Používanie jednoduchého konceptu subjektov a objektov umožnuje identifikovať základné operácie vykonávané v operačnom systéme. Bezpečnostný mechanizmus dokáže deterministicky rozhodnúť či daný subjekt má povolenie vykonať danú operáciu na danom objekte. Čo v skratke znamená, odpovedanie na otázky typu: *Má proces 87342 povolenie čítať súbor '/etc/passwd'*? Na vykonávanie podobných akcií je potrebné definovať tzv. bezpečtnostné politiky. Bezpečtnostná politika je súrhn pravidiel pre konkrétny subjekt.

Na Obrázku 2.2 je možné vidieť grafickú reprezentáciu rozhodovania.

Obrázek 2.2: MAC rozhodovanie

2.3 SELinux

SELinux [10] je *open-source* [2] produkt, implementujúci povinné riadenie prístupu(anglicky *Mandatory Access Control*, alebo *MAC*), čo umožňuje citlivejšie nastavenie prístupových práv. Hlavnou úlohou SELinuxu je zabezpečiť, aby každý proces mal svoj "sandbox[2]" [11], v ktorom bude mať povolené pracovať a zároveň aby nezasahoval do "sandboxu"iného procesu, pokiaľ mu to nie je povolené. SELinux je "aplikačný firewall", ktorý vymedzuje rozsah činnosti aplikáciam a službám, čím poskytuje vysoký stupeň bezpečnosti operačného systému.

SELinux obsahuje bezpečnostné politiky (množiny pravidiel popisujúce chovanie programu alebo služby), ktoré povolujú konkrétne chovanie daného programu, resp. služby. V zásade platí, že čo nie je povolené v bezpečnostnej politike je implicitne zakázané. Tieto bezpečnostné politiky sú administrátorsky konfigurovateľné, čo umožňuje pridávanie pravidiel podľa potrieb konkrétneho systému. Bezpečnostné politiky sú skompilované do binárnej podoby (tzv. moduly) a následne sú zavádzané do jadra operačného systému. Tieto moduly je možné dynamicky spravovať, teda pridávať a odoberať počas behu systému bez nutnosti reštartovania operačného systému.

Beh SELinuxu je možné rozdeliť na 3 režimy. Prvým je režim *Disabled*, kedy je SELinux úplne vypnutý.

Druhým je režim *permissive*, kedy SELinux jednotlivé pravidlá v politikách nevynucuje, tzn. nedefinované akcie v pravidlách povolí, ale tieto akcie sú zaznamenané do záznamov systémov ako AVC (anglicky *Access Vector Cache*) správy. Tento režim je často používaný pre riešenie problémov s politikami, alebo pri ich samotnom písaní.

Posledným a najúčinejším z hľadiska bezpečnosti režimom je *Enforcing* režim. Tento režim prísne vynucuje právidlá politík, a teda ktorá akcia nie je definovaná v pravidlách politík je zaznamenaná do systémových záznamov a je odopreté jej vykonanie. Tento režim je najbezpečnejší a používa sa štandardne.

V predchádzajúcej kapitole boli zadefinované pojmy, ktoré vedú k jednoznačnej identifikácií akcie v systéme. Aktuálna kapitola sa bude venovať **bezpečnostnému kontextu**, ktorým je možné konkrétne špecifikovať typ subjektu/objektu. Následne bude predstavená sada SELinux pravidieľ definujúca povolujúce pravidlá pre subjekty tzv. **bezpečnostná politika**.

[2]sandbox - karanténa vyhradzujúca procesu len určitú časť operácií v systéme

2.4 História projektu SELinux

Od začiatku sedemdesiatych rokov minulého storočia bola publikovaná konceptuálna architektúra, popisujúca viac úrovňový bezpečnostný model. Konceptuálna architektúra sa skladala z možnosti izolácie jednotlivých procesov, čím by bolo možné zadefinovať a kontrolovať, ktoré procesy môžu výkonávať konkrétne akcie v operačnom systéme. Takto by bolo možné vytvoriť viac úrovňovú hierarchiu medzi procesmi. Najväčší pokrok v oblasti izolácií procesov nastal počas devädesiatych rokov, kedy sa na vývoji podujala *NSA (U.S. National Security Agency)* spolu s *SCC (Secure Comptuting Corporation)*. V tejto fáze bola vytvorená flexibilná a silná architektúra s povinným riadením prístupu, pričom bol kladený dôraz na teoretické základy a charakteristiku tejto architektúry.

Neskôr, spolu s Univerzitou v Utahu bol vytvorený prototyp nazvaný *Flask* [13]. Po ďalšej spolupráci so spoločnosťami *Network Associates* a *MITRE*, ktoré implementovali túto architektúru do operačných systémov Linux. Táto práca bola zverejnená ako open-source projekt v decembri roku 2000. V súčastnosti je SELinux podporovaný viacerými Linuxovými distribúciami ako je Fedora, Red Hat Enterprise Linux, Debian alebo SUSE.

V súčastnosti na projekte SELinux aktívne pracujú spoločnosti Red Hat a Tresys spolu s NSA. SELinux je masívne nasadzovaný na malé i velké podnikové riešenia zabezpečenia operačného systému Linux, rovnako okolo tohto projektu pracuje aj početná komunita, čo dosvedčuje fakt, že SELinux je zapnutý automaticky po inštalácií jednej z najvačších komunitných distribúcií Linuxu Fedore.

2.5 SELinux kontext

SELinux kontext [18] (nazývaný aj bezpečnostný kontext) je základným prvkom pri riadení povinného prístupu v SELinuxe. Jedná sa o tzv. bezpečnostný atribút, ktorý je pridelený všetkým subjektom ako aj objektom v systéme. Je uložený v rozšírených atribútoch súboru. Pomocou bezpečnostného kontextu dokáže SELinux detekovať typ subjektu/objektu a podľa toho rozhodnúť o povolení alebo zablokovaní vykonávanej akcie. Bezpečnostný kontext sa skladá z troch častí, ktoré sú od seba oddelené dvojbodkou.
Príklad bezpečnostného kontextu:

system_u:system_r:init_t

- **Identita**

  system_u:system_r:init_t

  Identita určuje SELinux uživateľa priradeného k subjektu alebo objektu. V prípade, že sa jedná o subjekt, identita určuje pod akým SELinux uživateľom je proces spustený. V prípade objektu, určuje vlastníka. Je potrebné rozlišovať SELinux identitu a klasickú identitu uživateľa v Linuxe. SELinux identita môže byť pridelená viacerým užívateľom v systéme s rovnakými, prípadne podobnými oprávneniami.

- **Rola**

  system_u:**system_r**:init_t

  SELinux umožnuje uživateľovi vstúpit do jednej alebo viac rolí. Každá rola definuje sadu oprávnení, ktoré môže uživateľ získať. V danom časovom okamihu, uživateľ môže byť vždy len v jednej roli. Pričom uživateľ môže prechádzať medzi týmito rolami

pomocou príkazu *newrole*. SELinux definuje špeciálnu rolu, *sysadm_r*, používanú pre spravovanie SELinuxových zariadení.

- **Typ**

  system_u:system_r:**init_t**

  Typ sa tiež v prípade subjektu môže nazývať ako *Doména*. Je to základný bezpečnostný atribút SELinuxu, ktorý sa používa na rozhodovanie o povolení vykonaní akcie. Pomocou typov sú procesy vkladané do **sandboxov**, čím je zaručená izolácia procesov, a teda aj eskalácia privilégií. Je možné povedať, že **typ** je pomenovaný sandbox. V praxi používa SELinux len zopár identít a rôl, no v prípade typov sa ich použitie ráta na stovky v jednom systéme.

2.6 Bezpečnostná politika

Bezpečnostná politika [18] je konečná množina pravidiel, ktorými sa riadi SELinux. Definuje typy pre objekty a domény pre subjekty v systéme. Ďalej definuje role pre uživateľa, ktoré domény môžu vstúpit do inej domény. Aké typy su prístupné pre konkrétne domény a pod.

V minulosti sa používala jedna monolitická NSA SELinux politika. Všetky pravidlá boli kompilované do jedného binárneho súboru a akákoľvek zmena znamenala kompletné prekompilovanie všetkých pravidiel a opätovné zavedenie do jadra operačného systému. V súčastnosti sa používa mechanizmus nazývaný **LSM (Linux Security Modules)**, rozdelujúci monolitickú politiku na viac modulov. Každý proces, pracujúci v svojom sandboxe, má priradený svoj **modul**, ktorý je možné za behu zavádzať do jadra, prípadne ho odobrať bez zásahu iných modulov. Tento mechanizmus uľahčuje prácu s administráciou bezpečnostných politík, vytváranie určitej hierarchie a modularity.

Bezpečnostný modul

Bezpečnostný modul popisuje akceptovateľné správanie procesu, a tým ho izoluje do karantény. Bezpečnostný modul pozostáva z troch súborov a tie sú nasledovné:

- Privátna politika modulu
  Obsahuje deklaráciu a lokálne pravidlá modulu. Táto časť je obsiahnutá v *.te* súboroch.

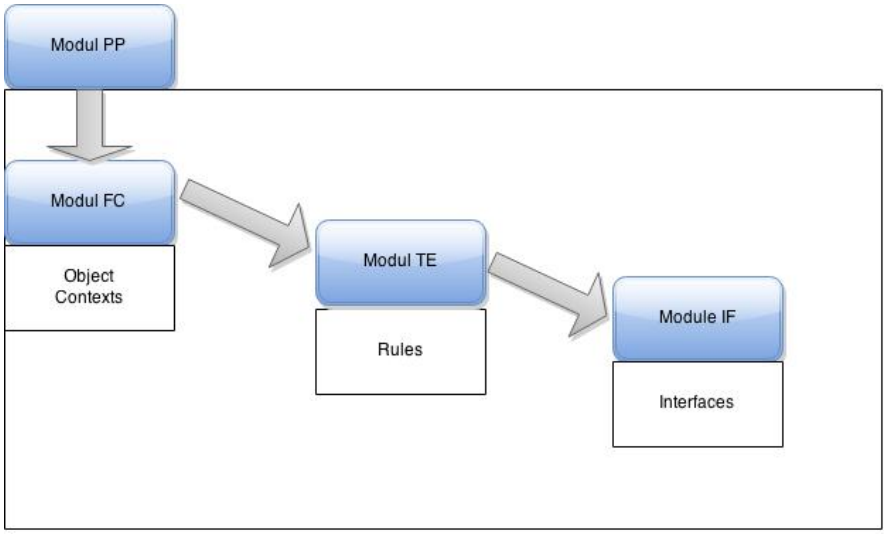
- Externá sada pravidiel
  Definuje makrá pre prácu externých domén s internými doménami alebo typmi v module. Táto časť je obsiahnutá v *.if* súboroch.

- Politika označovania súborov
  Definuje bezpečnostný kontext pre objekty, s ktorými privátne pracuje proces patriaci konkrétnemu modulu. Táto časť je obsiahnutá v *.fc* súboroch.

Kompiláciou týchto troch súborov je získaný binárny bezpečnostný modul, ktorý je možné zaviesť do jadra operačného systému. Na obrázku je zobrazený bezpečnostný modul zapúzdrujúci popisované súbory(viď. Obr. 2.3).

2.7 Auditovací systém

Audit [17] systém poskytuje v operačnom systém Linux spôsob, ako sledovať bezpečnostne-relevantné informácie o stave systému. Na základe vopred nakonfigurovaných pravidiel



Obrázek 2.3: Bezpečnostný modul

služba auditd generuje záznamy s informáciami o udalostiach, ktoré sa dejú v systéme. Takýto typ informácií je kľučový pre detekovanie nesprávneho fungovania systému alebo detekovania možného útoku na systém. Služba auditd ďalej neposkytuje žiadne bezpečnostné mechanizmy systému, je primárne určená k objavovaniu porušenia bezpečnostných pravidiel v systéme. Týmto porušeniam môže napríklad zabrániť bezpečnostný mechanizmus ako je SELinux.

Nasledujúci zoznam sumarizuje informácie, ktoré je možné pomocou služby auditd uchovávať v záznamoch:

- Dátum, čas, typ a výsledok udalosti
- Identita uživateľa, ktorý vyvolal udalosť
- Pokus o modifikáciu konfiguračných súborov ako je napr. */etc/passwd*
- Neautorizovaná udalosť z pohľadu SELinuxu.

V prípade, že dôjde k vyššie spomínanej udalosti a služba auditd detekuje neautorizovaný prístup, ktorý bol detekovaný SELinuxom, je generovaná správa AVC.

AVC správa

Popis správy AVC je vysvetlený na nasledujúcom príklade:
type=AVC **msg**=audit(1425818718.415:1124): avc: **denied** { write } for **pid**=6476 **comm**="mysqld"**name**="mariadb.pid"**dev**="tmpfs"**ino**=540005 **scontext**=system_u:system_r:mysqld_t:s0 **tcontext**=system_u:object_r:var_run_t:s0 **tclass**=file **permissive**=0

Vygenerovaná správa je uložená v súbore: */var/log/audit/audit.log.*

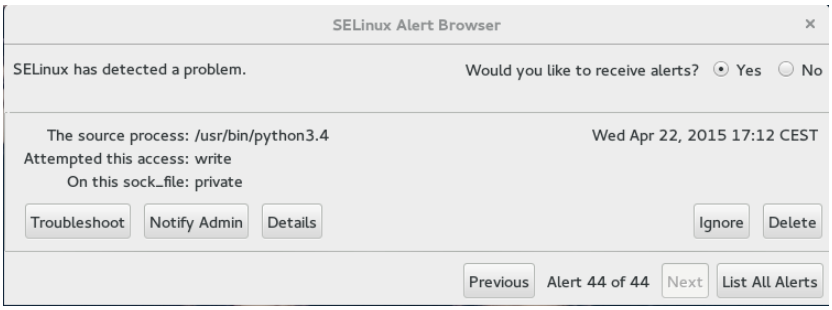
Časť AVC správy	Popis
type	Typ audit správy je AVC
msg	*timestamp*[3] správy
denied	Informácia čo SELinux urobil (napr. odmietnutie činnosti)
pid	Identifikátor procesu, ktorý bol zablokovaný SELinuxom
comm	Názov procesu, ktorý bol zablokovaný SELinuxom
name	Názov, prípadne cesta k objektu
dev	Zariadenie, na ktorom je umiestnený objekt
ino	Inode číslo cieľového súboru
scontext	Zdrojový SELinux kontext
tcontext	Cieľový SELinux kontext
tclass	Trieda cieľového súboru

Tabulka 2.3: Popis AVC správy

2.8 Setroubleshoot

Systém Setroubleshoot [19] umožňuje užívateľovi oznámenie, diagnostikovanie a podanie uživateľsky prívetivé vysvetlenie, prečo k danému zablokovaniue služby prišlo. V niektorých prípadoch Setroubleshoot dokáže dodať odporúčania ako systém prispôbiť tak, aby k podobnej situácii už nedošlo v budúcnosti. Systém Setroubleshoot pozostáva z dvoch komponent:

- Serverová časť **Setroubleshootd**
- Uživateľská časť **Sealert**



Obrázek 2.4: Sealert - základné informácie

Setroubleshootd je systémový daemon pracujúci pod právami superuživateľa a načúvajúci audit udalostiam emitovaných z jadra v súvislosti s bezpečnostným mechanizmom SELinux. V prípade, že Setroubleshootd daemon detekuje SELinux AVC správu, vykoná nad ňou sériu analýz pomocou pluginov, ktoré skúmajú audit data prislúchajúce k danej AVC správe. Výsledok analýzy je zaznamenaný a je zaslaný signál klientskej časti o novom zázname v setroubleshootd databázi.

Sealert môže fungovať v dvoch režimoch, v grafickom a textovom. V oboch prípadoch proces beží pod právami konkrétneho uživateľa. V grafickom režime čaká na nové záznamy zo serverovej časti, ak je detekovaný nový záznam je uživateľ o ňom oboznámený pomocou notifikácie(viď. Obr. 2.3 a 2.4). Uživateľ môže kliknúť na notifikáciu, ktorá otvorí nové okno

s popisom záznamu. Okrem aktuálneho záznamu môže uživateľ prehľadávať aj predchádzajúce záznamy, uložené v Setroubleshootd databáze. V prípade textového režimu, uživateľ musí sealert spustiť a skontrolovať či nedošlo k novým Setroubleshoot záznamom. Ak k záznamom došlo, uživateľovi sú ponúknuté rovnaké akcie ako v prípade grafického režimu.

Obrázek 2.5: Sealert - detailné informácie

Kapitola 3

Správa záznamových správ

Operačné systémy sa počas vývoja stávali čoraz viac komplexné a zložité systémy, pri ktorých nebolo možné sledovať všetku činnosť v reálnom čase. Preto bolo potrebné vyvinúť spôsob uchovávania záznamov o systéme, či už sa jednalo o správy informatívneho charakteru alebo správy o zlyhaní alebo havarovaní služby. Jednou z popredných výhod operačných systémov typu Linux je možnosť uchovávania akejkoľvek akcie, ktorá je zaznamenaná. Táto možnosť je neoceniteľná pri snahe detekovať alebo analyzovať co sa práve v systéme deje.

Nahliadnutie do logov operačného systému by malo byť prvým krokom aj pri zistení, že akákoľvek služba nepracuje správne alebo zhavarovala. V tejto časti budú spomenuté spôsoby a dôvody uchovávania logov, ich analýza a prípadné riešenia ako informovať vývojárov danej služby o chybách pomocou práve systémových logov.

3.1 Záznamové správy

Záznamová správa v logoch je najmenší prvok, ktorý dokáže podať ucelenú informáciu o konkrétnej akcií, ktorá sa udiala v minulosti. V prípade že sa jedná o neštrukturovanú záznamovú službu, jedna správa obsahuje jeden riadok, naopak pri štrukturovaných službách je často používaný systém ukladania spôsobom kľúč - hodnota.

Záznamové správy sú ukladané v súboroch, aby bola zaručená perzistencia dát aj pri reštartovaní operačného systému. Najčastejšie sú zaznamové správy ukladané do adresára */var/log/*, ktorý je určený práve pre tieto účely. Samotné súbory logov sú ďalej rozdelené na viac súborov z dôvodu ľahšieho vyhľadávania. V Linuxovej distribúcií Fedora adresár */var/log/* obsahuje súbory ako je napr. *boot.log, kdm.log, yum.log, messages.* Je potrebné spomenúť, že ako adresár, tak aj názvy súborov sa môžu líšiť v závislosti na konkrétnej distribúcií Linuxu. Ukladanie logov nemusí byť vždy nasmerované do súborov, správy môžu byť posielané na iný systém, ktorý bude tieto správy skladovať. Výhodou skladovaniu logov mimo samotný systém je fakt, že v prípade úspešného napadnutia systému útočníkom môže dôjsť k úmyselnému odstráneniu logov pri zahladzovaní stôp. Pokiaľ však budú logy uložené na inom systéme, útočník ich vymazať nemôže, a teda ani zamaskovať útok.

Dôležitým pojmom v prípade logovania je tzv. **rotácia** logov. Nakoľko počas behu systému logovací systém narastá je potrebné ich priebežné odstraňovanie. Rotácia spôsobí, že po určitom definovanom čase sa logy začnú odstraňovať z logovacích súborov. Na tieto účely slúží nástroj **logrotate**.

Vačšina služieb, ktoré slúžia práve na zaznamenávanie správ o stave služieb alebo samot-

neho jadra operačného systému neanalyzujú príslušné logy. Analýza zostáva na užívateľovi, prípadne môže existovať program na automatickú analýzu správ.

3.1.1 Syslog

Logovacia služba Syslog [15] bola vyvíjaná v 80. rokoch minulého storočia ako časť projektu Sendmail[1]. Neskôr sa ukázal ako hodnotná služba aj mimo logovanie len služby Sendmail a tak sa stal štandardným riešením pre zaznamenávanie správ pre operačné systémy Unixového typu. Časom vzniklo mnoho derivátov implementácie syslogu, napr. pre aktívne sieťové zariadenia ako sú smerovače.

Syslog funguje na základe čítania prichádzajúcich správ zo schránky[2] */dev/log* a na základe konfigurácie, ktorá sa nachádza v súbore */etc/syslog.conf*, ich filtruje a ukladá ako záznamové správy do súboru */var/log/messages*.

Jednou z výhod služby Syslog je kategorizovanie správ a rozdelenie podľa priority [5]. Priorita definuje ako je správa dôležitá, k dispozícií je 9 rôznych priorít. Atribút kategórie hovorí o oblasti, z ktorej správa pochádza. K dispozícií je 12 zadefinovaných správ.

Zaznamenávanie správ pomocou syslogu je jednoduché. Knižnica **libc** poskytuje jednoduché rozhranie pre zápis do syslogu. Pozostáva z funkcií obsiahnutých v tabuľke 3.1.

Funkcia	**Popis**
openlog()	Otvorenie logu
syslog()	Zápis do logu
closelog()	Zatvorenie logu

Tabulka 3.1: Funkcie syslogu

V prvom kroku zápisu do syslogu je zavolaná funkcia *openlog()* a je zvolená kategória správy. Potom je možné zapisovať do syslogu volaním funkcie *syslog()*, parametrom tejto funkcie je priorita aktuálnej správy a samotný text správy. Na konci je odosielanie do syslogu uzatvorené zavolaním funkcie *closelog()*.

Nevýhody používania syslogu

Syslog obsahuje niekoľko nevýhod, z tohto dôvodu je v súčastnosti nahradzovaný službou journald, ktorá je popísaná v nasledujúcej kapitole. Prvou nevýhodou je konzistencia. Protokol syslog nedefinuje žiadny štandardný spôsob ako formátovať prichádzajúce správy. Formát je definovaný vývojárom služby alebo aplikáciu, ktorá zapisuje do syslogu. Z čoho vyplíva, že niektoré správy sú čitatelné pre človeka, niektoré nie sú. Syslog len poskytuje spôsob ako dopraviť správu.

V prípade logovania správ na vzdialený systém syslog používa na transportnej vrstve protokol UDP, s čím sú spojené známe problémy, nakoľko je UDP protokol nespoľahlivý. Môže teda dôjsť k strate záznamových správ z dôvodu preťaženia siete alebo straty packetov.

Ďalej v syslogu neexistuje žiadna **autentizácia**, čo znamená že akýkoľvek systém môže poskytnúť falošnú identitu, a teda môže aj podstrčiť falošné záznamové správy. Taktiež je náchylný na útoky typu DOS resp. DDOS.

[1] Služba, zaistujúca odosielanie elektronickej pošty
[2] socketu

Poslednou a významnou nevýhodou je fakt, že syslog nedokáže zaznamenávať a ukladať správy v štrukturovanej podobe. Každá správa je zobrazená ako reťazec, čo znemožňuje administrátorom filtrovanie a vyhľadávanie správ podľa kľúča alebo podľa hodnoty v danom kľúči. Tento nedostatok nahradzuje služba **journald**.

3.1.2 Systemd-journald

Služba journald je súčasťou revolučného systemd. Systemd je sada služieb, knižníc a nástrojov pre centrálnu správu a konfiguráciu systému vzhľadom na architektúru pre operačné systémy Linux. Spúšťa sa s idendifikačným číslom 1. Nakoľko je logovacia služba súčastou tejto sady je možné zaznamenávanie správ od úplného začiatku štartu systému, čo je výhoda oproti logovacej službe syslog, ktorá je spustená až v neskorších fázach štartu systému.

Logovacie data sú zbierané, ukladané a spracovávané pomocou žurnálov služby journald. Táto služba ich vytvára a udržuje v binárnej podobe. Do žurnálov sú vkladané informácie z jadra operačného systému, užívateľských procesov, štandardného výstupu a štandardného chybového výstupu. Tieto údaje sú vkladané pomocou natívneho API.

Dôležitou zmenou oproti službe syslog je fakt, že journald ukladá informácie v štrukturovanej podobe. Štruktúry sú podobné asociatývnym poliam, ktoré sú rovnako prezentované spôsobom: **kľúč = hodnota**. Na obrázku 3.1 je príklad správy z journald.

```
Fri 2015-04-24 14:28:00.729806 CEST
    PRIORITY=6
    _UID=0
    _GID=0
    _BOOT_ID=9a4606e433df4065a790b4ec21d7746f
    _MACHINE_ID=9ac823f0af654d86beea3e30b5744e9b
    _HOSTNAME=Thinkpad
    SYSLOG_FACILITY=3
    SYSLOG_IDENTIFIER=systemd
    CODE_FILE=../src/core/job.c
    CODE_LINE=732
    CODE_FUNCTION=job_log_status_message
    MESSAGE_ID=39f53479d3a045ac8e11786248231fbf
    RESULT=done
    _TRANSPORT=journal
    _PID=1
    _COMM=systemd
    _EXE=/usr/lib/systemd/systemd
    _CAP_EFFECTIVE=3ffffffff
    _SYSTEMD_CGROUP=/
    _CMDLINE=/usr/lib/systemd/systemd --switched-root --system --deserialize 22
    _SELINUX_CONTEXT=system_u:system_r:init_t:s0
    UNIT=systemd-readahead-collect.service
    MESSAGE=Started Collect Read-Ahead Data.
    _SOURCE_REALTIME_TIMESTAMP=1429878480729806
```

Obrázek 3.1: Príklad journald správy

Pomocou štruktúrovanej serializácie údajov je umožnené jednoduché a rýchle filtrovanie a vyhľadávanie správ. Taktiež je možné do jednej správy zapúzdriť oveľa viac informácií ako v prípade syslogu, kde je možné všetky informácie uložiť len ako dlhý reťazec. Na priloženom obrázku je možné vidieť, že journald prijal správu o štarte služby PakcageKit. Táto informácia je uložená pod kľúčom *MESSAGE*. No zároveň boli pridané doplňujúce informácie, ktoré boli vygenerované službou systemd. V prípade potreby môže vývojár alebo uživateľ pridávať hodnoty, do kľúčov prípadne vytvárať vlastné dvojice kľúč - hodnota.

V dobe písania práce je journald, ako súčasť projektu systemd, masívne nasadzovaný na najpopulárnejšie distribúcie operačného systému Linux a je nahradzovaný za staršiu logovaciu službu syslog.

3.2 Hlásenie chýb

V prípade, že logovací systém detekuje bezpečnostný útok alebo chybu, ktorá nie je napraviteľná administrátorom systému[3], je nutné túto chybu alebo zraniteľnosť oznámiť vývojárom. Správne a funkčné spôsoby nahlasovania chýb v programoch sú veľmi dôležité pre obe strany, ako pre zákazníka, tak aj pre spoločnosť poskytujúcu daný produkt. Zákazník potrebuje chybu oznámiť, dostať sa čo najrýchlejsie k opravenej verzií a spoločnosť potrebuje informovať a opraviť všetky chybné verzie v čo najkratšom čase. Rozličné spoločnosti alebo vývojárske skupiny používajú rôzne formy nahlasovania chýb od tých základných (používanie elektronickej pošty) až po prepracované automatizované nahlasovanie chýb(napr. Automatic Bug Reporting Tool používaný spoločnosťou Red Hat). Nasledujúce kapitoly sa venujú dvom formám, a to reportovaniu na webové portály danej spoločnosti(produktu) a používanie automatizovaného nástroja na nahlasovanie chýb. Obe formy majú svoje výhody a nevýhody, ktoré sú prebrané nižšie.

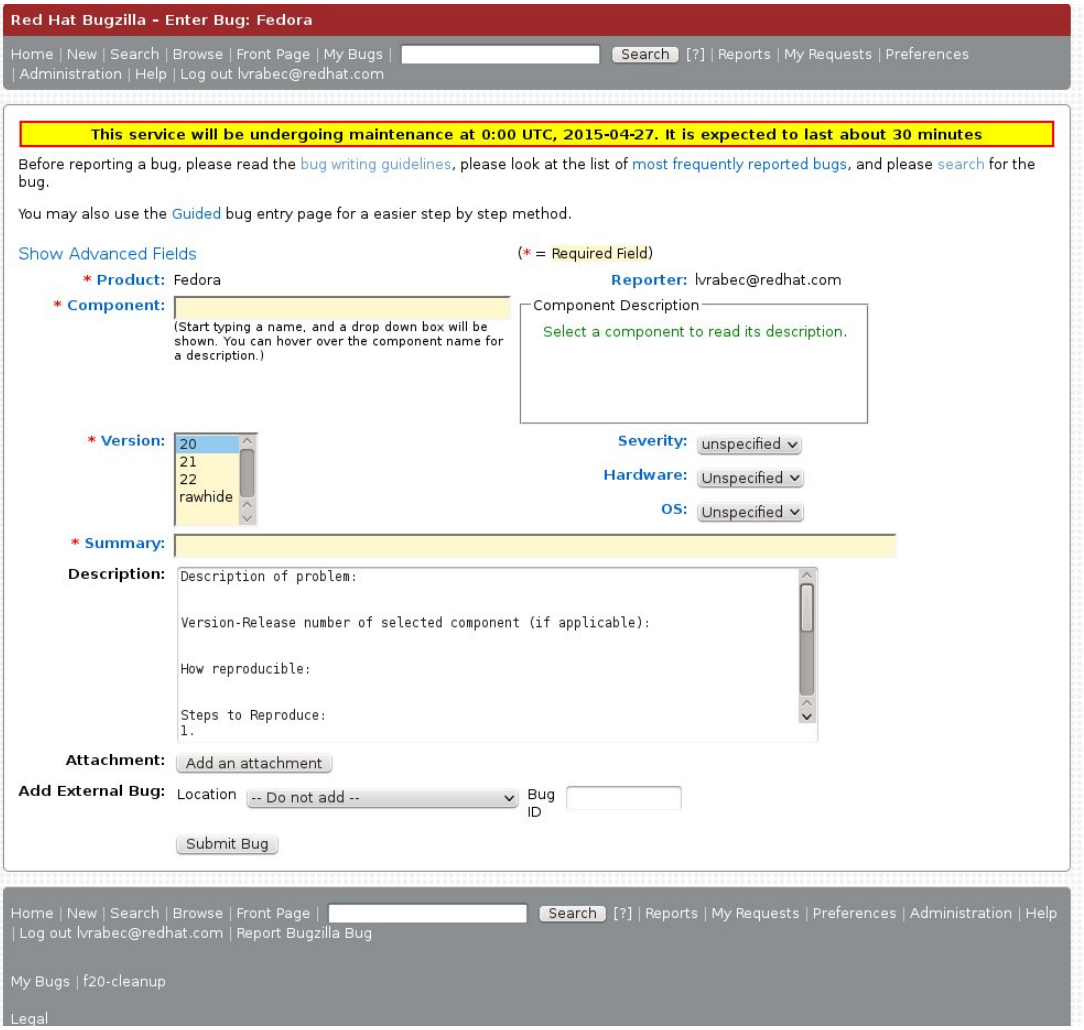
3.2.1 Bugzilla

Jednou z najstarších a najpoužívanejších spôsobov ako nahlásiť chybu je použitie webového informačného systému **bugzilla**. Tento spôsob je používaný aj pre produkty spoločnosti Red Hat. Ak dôjde k chybe a administrátor systému potrebuje zareportovať nový bug[4], prihlási sa na webový portál bugzilly a vyplní potrebné údaje o chybe. Údaje sú nasledujúce (postup bude prezentovaný na portáli Red Hat bugzilla(viď. obr. 3.2)):

- *Vybratie produktu*
  Adminitrátor si vyberie produkt, v ktorom prišlo k chybe, napr. *Fedora*
- *Komponenta produktu*
  Konkretizovanie časti produktu, ktorá obsahuje chybu, napr. *Selinux-policy*
- *Verzia produktu*
  Verzia produktu používaná na systéme, napr. *verzia 21*
- *Severity*
  Označenie vážnosti chyby, ktorá je nahlasovaná. Može nadobúdať hodnôt: urgetná, vysoká, stredná a nízka.
- *Hardware*
  Definuje architektúru, na ktorej došlo k chybe, napr. *x86_64*
- *OS*
  Operačný systém, na ktorom nastala chyba, napr. *Linux*
- *Zhrnutie*
  Popis chyby jednou vetou, ktorá bude použitá ako nadpis pre daný report, napr. *SELinux is preventing logrotate from 'create' accesses on the file hawkey.log*
- *Prílohy*
  Do prílohy je možné vložiť textové súbory, napr. samotné logy, ktoré oznámili chybu.

[3] Chyba je v aplikácií nie v konfigurací danej služby
[4] chybu

- *Popis*
  V popise je nutné podrobnejšie rozobrať chybu. Tu je možné uviesť informácie ako sú početnosť opakovania chyby a ci je chyba zreprodukovateľná. V prípade, že chyba je zreprodukovateľná je potrebné pridať presný postup v krátkych krokoch. Dôležité je aj popísanie aktuálneho(chybného) stavu a očakávaného(funkčného) stavu, ktorý ma byť obsiahnutý v novej verzií komeponenty alebo samotného produktu.



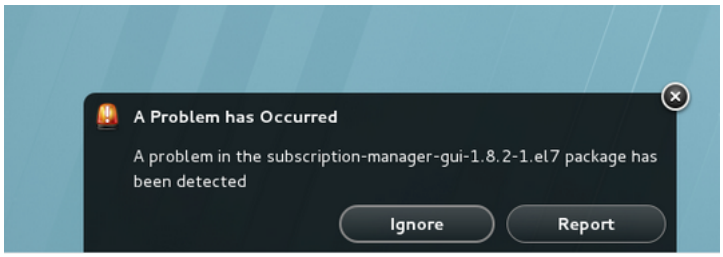
Obrázek 3.2: Red Hat Bugzilla

Ako je možné vidieť na predchádzajúcom odstavci, použitie webového informačného systému **bugzilla** je mierne zdĺhavé a pre neskúseného užívateľa alebo administrátora aj neprehľadné a komplikované. Všetky potrebné informácie musí užívateľ poznať a zadať ich manuálne. Počas týchto krokov môže dôjsť k viacerým chybám pri zadávaní informácií, či z dôvodu nepozornosti alebo neznalosti. To vedie k reportovaniu nepresných a fiktívnych chýb. Ďalším problémom je samotný zdĺhavý proces, ktorý mnoho užívateľov odradí od nahlásenia chyby, pokiaľ sa nejedná vyslovene o urgetný problém.

3.3 ABRT

Ako bolo spomenuté, proces nahlásenia chyby musí byť jednoduchý a krátky pre užívateľa nahlasujúceho chybu. Zároveň musí byť správny a presný pre vývojára, ktorý nahlasenie o chybe obdrží. Z dôvodov, ktoré boli spomenuté v predchádzajúcej kapitole, bol vyvinutý spoločnosťou Red Hat automatizovaný nástroj **ABRT** na hlásenie chýb.

ABRT [1] sa skladá z démona, ktorý monitoruje záznamové správy, zrútenia systému a spúšťa rôzne udalosti o zbere dát na základe typu chyby v systéme. Taktiež poskytuje okamžité informácie uživateľovi o chybe pomocou ikony na notifikačnej lište(viď. Obr. 3.3) grafického rozhrania operačného systému.



Obrázek 3.3: ABRT notifikácia

Ak dôjde k chybovému stavu v systéme, nástroj ABRT chybu detekuje v záznamových správach, začne zhromažďovať potrebné data, ako sú logovacie záznamy, história príkazov a podobne. K detekcií chýb dochádza pomocou sledovacích nástrojov, kedy ABRT démon sleduje systémové správy, D-BUS signály [8], prípadne je o chybách upozornený pomocou ABRT API naprogramovaného v programovacom jazyku Python [16]. Po tomto úkone začne podnikať potrebné kroky na nahlásenie chyby v závislosti od typu chybového stavu systému, nastavenia konfigurácie v konfiguračnom súbore abrt.conf uloženého v adresáry */etc/*. K dispozícií sú pluginy pre rôzne akcie, ktoré má ABRT vykonať pri detekcii chyby. Príklady použitia pluginov: nahlásenie havárie služby alebo aplikácie na webový portál bugzilla pomocou elektronickej pošty, odoslaním automatického e-mailu s vygenerovaným popisom chyby alebo prenos správy na FTP[5] server.

V prípade nahlasovania chýb pomocou nástroja ABRT sú všetky potrebné informácie zozbierané automaticky a uživateľ nemusí manuálne nič vypĺnať, čím je zamedzené podaniu nesprávnej informácie vývojárovi. Pomocou **ABRT** nástroja je možné odoslať systémovú chybu, no doposiaľ nebola implementovaná funkcia, ktorá by umožnovala detekovať a nahlásiť potencionálne nekorektne zadefinované bezpečnostné SELinux politiky.

[5] File Trasnfer Protokol

Kapitola 4

Analýza

V Linuxových distribúciach Fedora a Red Hat Enterprise Linux existuje viac spôsobov ako oboznámiť uživateľa operačného systému o odmietnutí vykonania akcie pomocou bezpečnostného mechanizmu SELinux. Týmito spôsobmi sú napríklad textové nástroje pre zobrazenie AVC správ, nahliadnutie do logov operačného systému. Tieto možnosti ale neupozornia uživateľa okamžite, ale až v prípade, ak uživateľ zaregistruje nesprávne fungovanie systému. Z týchto dôvodov bol implementovaný SEtroubleshoot systém, ktorý okamžite oznámil prítomnosť novej AVC správy, a teda informácií o odopretí akcie.

Problémom je viac možných rozhraní nahlasujúcich správy o systémových chybách. Linuxové distribúcie Fedora a RHEL[1] používajú centralizovaný "crash systém"Automatic bug reporting tool, ktorý okamžite oboznamuje uživateľa pomocou notifikácií v systémovej lište. Všetky záznamy o havarovaní jadra, systémových služieb a uživateľských procesov sú zobrazené pomocou nástroja ABRT. Záznamy o odopretí akcie pomocou mechanizmu nástroja SELinux sú zobrazené nástrojom SEtroubleshoot. Tento stav je nežiadaný, nakoľko existujú dve grafické uživateľské rozhrania. Preto je potrebná integrácia SELinux AVC správ o odopretí akcie do nástroja ABRT. Tak bude podporované len jedno rozhranie.

Taktiež prehľadnosť logovacích správ nie je ideálna a je potrebná analýza pre prípadne možnosti zlepšenia podávania informácií týmto spôsobom.

Úlohou teda bude vytvoriť systém, ktorý by dokázal spracovávať AVC správy, podrobiť ich analýze a následne oznamovať užívateľovi pomocou nástroja ABRT ich vzniknutie. Tento nástroj by dokázal podať riešenie problému prípadne možnosť odoslať túto správu zodpovedným vývojárskym skupinám spoločnosti Red Hat.

V tejto kapitole budú prezentované dôvody a možné zlepšenia upozorňovania uživateľov o nových AVC správach. Tieto analýzy budú konzultované na pravidelných stretnutiach spolu s tímami spoločnosti Red Hat, ktoré sa priamo podielajú na vývoji bezpečnostného mechanizmu SELinux, automatizovaného nástroja na hlásenie chýb ABRT a logovacej služby journald.

4.1 Požiadavky

Hlavnou požiadavkou je centralizovanie záznamov o detekcií AVC správ generovaných auditd službou pomocou nástroja ABRT. V prípade že dôjde k takejto detekcií je potrebné uživateľa rýchlo a jednodnucho informovať a aktuálnom stave systému a o tom čo sa udialo. Užívatelovi budú poskytnuté základné informácie o udalosti a službe, ktorá sa pokúšala

[1] Red Hat Enterprise Linux

o neutorizovanú akciu. V prípade že uživateľ bude vyžadovať viac informácií, je potrebné podať podrobné detaily.

Eventuálnym požiadavkom je úprava grafického uživateľského rozhrania nástroja ABRT pre potreby správneho a prehladného zobrazovania AVC správ.

Ďalším požiadavkom je automatizovaná analýza správy AVC, predtým ako je vôbec oznámená uživateľovi. Pri automatizovanej analýze sú kladené požiadavky na rýchlosť analýzy z dôvodu čo najmenšieho zaťaženia systému. Ak výsledkom tejto analýzy je detekovanie zlej konfigurácie bezpečnostného mechanizmu SELinux alebo zlé definovanie pravidiel v bezpečnostných politikách, je podaný užívatelovi postup o prípadnej náprave tohto problému. Kroky vedúce k náprave musia byť riadne okomentované a nesmú viesť k vyvolaniu inej AVC správy.

Ak bude automatizovaná analýza neúspešná, uživateľovi budú zobrazené len informácie nachádzajúce sa v AVC správe a bude nutná manuálna analýza samotným uživateľom systému. Po manuálnej analýze budú poskytnuté dve možnosti. Prvá možnosť bude ignorovanie takejto správy a uživateľ nebude naďalej infomovaný o rovnakej správe v budúcnosti. Tento požiadavok je vyhovujúci, ak skúsený uživateľ vie čo sa deje so systémom, a napríklad sa snaží o úmyselné vyvolanie AVC správy, napríklad z dôvodu ladenia aplikácie, ktorej bola odopretá služba.

Najdôležitejším požiadavkom pri detekcií novej AVC správy je možnosť odoslania všetkých potrebných údajov a samotnej AVC správy vývojárskej skupine, ktorá sa bude zaoberať manuálnou analýzou tejto správy. V takomto prípade, ak uživateľ nevie čo sa udialo v systéme a nie je si istý, čo správa znamená, môže ju odoslať ako zápis o chybe. Správcovia bezpečnostných politík tento zápis manuálne zanalyzujú. Výsledkom môže byť detekcia útoku, ktorému SELinux zabránil alebo absencia chýbajúcich povolovacích pravidiel v bezpečnostných politikách.

Z predpokladu potencionálneho budúceho integrovania AVC záznamov do iných nástrojov bol zadefinovaný požiadavok aj pre vytvorenie jednotnej databázy, pre ukladanie AVC záznamov pred tým ako budú spracovávané koncovou aplikáciou (ako je napríklad momentálne nástroj ABRT). Tak bude zaručená možnosť budúcej integrácie s inými nástrojmi.

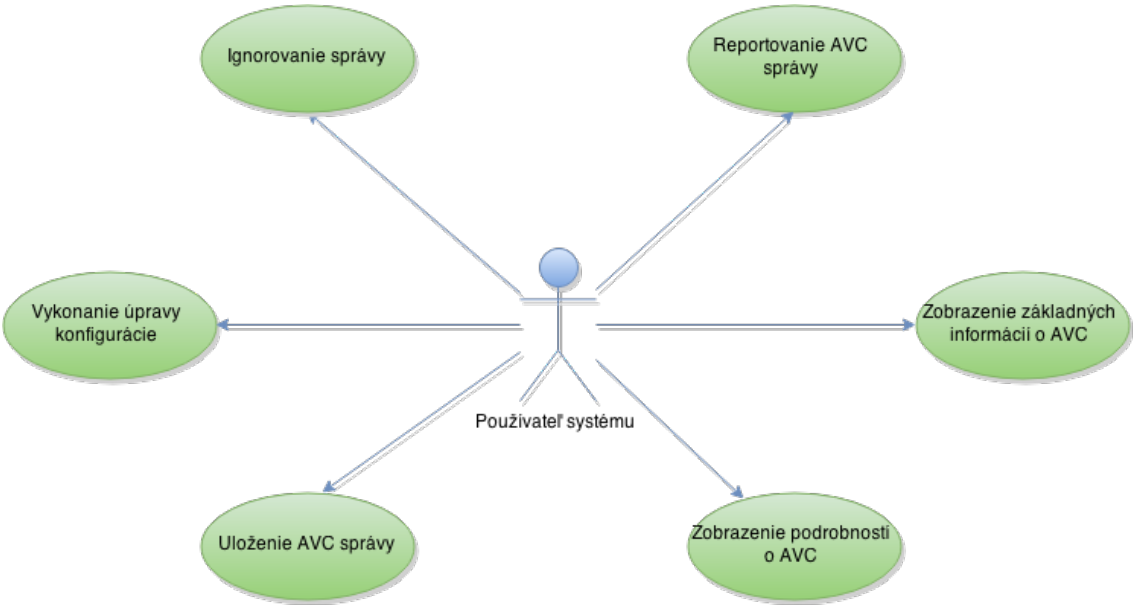
Posledným požiadavkom je možnosť ukladania AVC správ do internej databázi nástroja ABRT, aby bolo možné k AVC správam pristupovať aj v budúcnosti alebo po reštartovaní operačného systému.

Grafické zobrazenie diagramu prípadov použitia je možné vidieť na obrázku 4.1.

4.2 Aktuálny stav

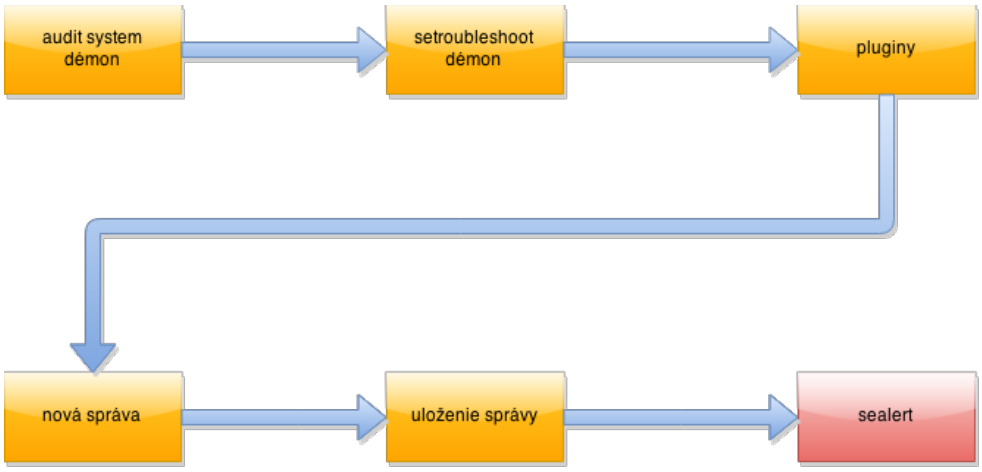
V súčastnosti existuje jediný spôsob o informovaní uživateľa pri generovaní AVC záznamu. Tou možnosťou je notifikácia a následné zobrazenie správy systémom setroubleshoot, konkrétne klientskou častou SEalert. Tento stav nie je prispôsobený na integráciu AVC záznamov do inej aplikácie ako je práve SEalert. Nástroj SEalert ale pri istých požiadavkách funguje správne a stabilne preto bude možné prevziať niektoré časti a znovu ich použiť pri budúcej integrácií. Nevýhodou SEalertu je fakt, že informácie o AVC správe sú podávané v neprehľadnej forme a mnoho užívateľov tieto notifikácie ani nečítajú, z čoho vyplýva, že nástroj nie je tak efektný ako môže byť.

Na Obrázku 4.2 je možné vidieť komplektný tok AVC správy. Audit démon vygeneruje správu AVC s popisom o odoprení služby, ktorá je odosláná do setroubleshoot démona



Obrázek 4.1: Diagram prípadov použitia v nástroji ABRT

pomocou technológie D-BUS[2]. Setroubleshoot démon prevedie sériu analýz daného AVC. Úlohou pluginov je nájsť známe problémy, ktoré sú na strane uživateľa a ako výsledok podať riešenie tejto chyby(napr. zlý kontext objektu). Zabalená AVC správa, už s prípadným riešením, je uložená v iternej databáze systému setroubleshoot. Následne je opäť pomocou technológie D-BUS [8] upozornená klientská časť sealert o novej správe v internej databáze. Táto správa je zobrazená v notifikačnej lište pomocou nástroja sealert.



Obrázek 4.2: Existujúce riešenie

[2]D-BUS je zbernicový systém odosielania správ, ktorý slúži na medziprocesovú komunikáciu.

Dôležitým problémom je aj neexistujúca možnosť sledovania AVC záznamov v logovacej službe journald. Táto absencia momentálne znemožňuje vytvorenie nového alebo integrovanie existujúceho riešenia pre notifikáciu nových AVC správ.

Z viacerých hladísk je tento stav nežiadúci, nakoľko je potrebné, priam až nutné integrovanie SELinux AVC správ do rozličných nástrojov, ktoré dokážu upozorniť či užívateľa pracovnej stanice alebo administrátora serveru. Vytvorenie centralizovanej databázy s novo vygenerovanými AVC správami by dokázalo tento problém riešiť.

4.3 Záver analýzy

Počas analýzy boli prediskutované jednotlivé časti systému setroubleshoot, jeho funkcionalita a prípadné nedostatky so "SELinux Solutions"tímom, ktorý mi poskytol priestor pre otázky a možné návrhy na zlepšenie. Rovnako boli prediskutované aj časti nástroja ABRT, princíp tvorby ABRT modulov na zber dát a prípadné riešenia úpravy grafického uživateľského rozhrania spolu s ABRT tímom v spoločnosti Red Hat. Počas fázy analýzy projektu boli konzultované možnosti logovacieho nástroja journald so systemd tímom, ktorý navrhol prípadné vylepšenia týkajúce sa použitia nástroja journald ako databázu pre AVC správy. Boli prehodnotené existujúce riešenia, ktoré budú znovu použité pri návrhu a implementácií integrácie.

Záverom analýzy sú funkčné požiadavky, na ktoré sa bude brať dôraz pri návrhu integrácie.

4.4 Nasledujúce kroky

V tomto kroku je možné prehlásiť fázu životného cyklu analýzy projektu za ukončenú. Funkčné požiadavky sú jasne definované, rovnako aj diagram prípadov použitia pre nástroj ABRT. Je možné že k prípadnej neskoršej analýze istých častí dôjde v budúcnosti. Pomocou skutočností zistených v tejto kapitole bude vytváraný návrh integrácie.

Kapitola 5

Návrh implementácie

V nasledujúcej kapitole bude popísaný a podrobne rozobratý návrh samotnej integrácie projektu do existujúceho riešenia. Táto kapitola vychádza z predchádzajúcej časti - Analýzy integrácie, pomocou ktorej boli zohľadnené možné problémy pri návrhu projektu. Návrh aplikácie bude postupne rozdelený na viac ucelených logických celkov, vďaka ktorým bude samotný návrh lepšie pochopitelný. Bude zdôvodnená architektúra projektu, ktorá bola už predstavená pri obhajobe semestrálneho projektu. Pri návrhu sa dbalo na použitie aktuálnych technológií a možnosti rýchleho a jednoduchého implementovania ďalších nástrojov na hlásenie chýb.

Po návrhu integrácie projektu sa táto kapitola zameriava aj na návrh grafického užívateľského rozhrania v nástroji ABRT.

Posledným krokom pri návrhu bude zhrnutie a zadefinovanie najbližších krokov potrebných pre implementáciu.

5.1 Proces návrhu

Proces návrhu vznikal bezprostredne po zadefinovaní funkčných požiadaviek, ktoré boli vytvorené v procese analýzy integrácie. Následne boli navrhnuté technológie, ktoré je možné na tieto účeli použiť. Každá navrhnutá technológia bola preskúmaná a prediskutovaná či je vhodná na potreby integrácie v tejto práci. Po vybratí technológií bol vytvorený predbežný grafický návrh.

Počas procesu návrhu bol spočiatku vybratý návrh, ktorý sa neskôr ukázal ako nevhodný pre takýto typ integrácie. Jeho princíp spočíval v tom, že všetká komunikácia medzi systémom setroubleshoot a ABRT prebiehala pomocou technológie D-BUS. D-BUS je zbernicový systém odosielania správ, ktorý slúži na medziprocesovú komunikáciu. Jedná sa o pokročilejšiu nadstavbu BSD schránok. Tento návrh sa ukázal ako nevhodný z dôvodu, že pri enormnom odosielaní správ setroubleshootd deamona na systém D-BUS nedochádzalo k úplnemu obdržaniu množstva správ, čo znamenalo že nástroj ABRT neobdržal všetky správy. Ďalšou nevýhodou bolo neuložnie AVC správy do logovacej služby, z ktorej by bolo možné AVC prečítať z akejkoľvek aplikácie.

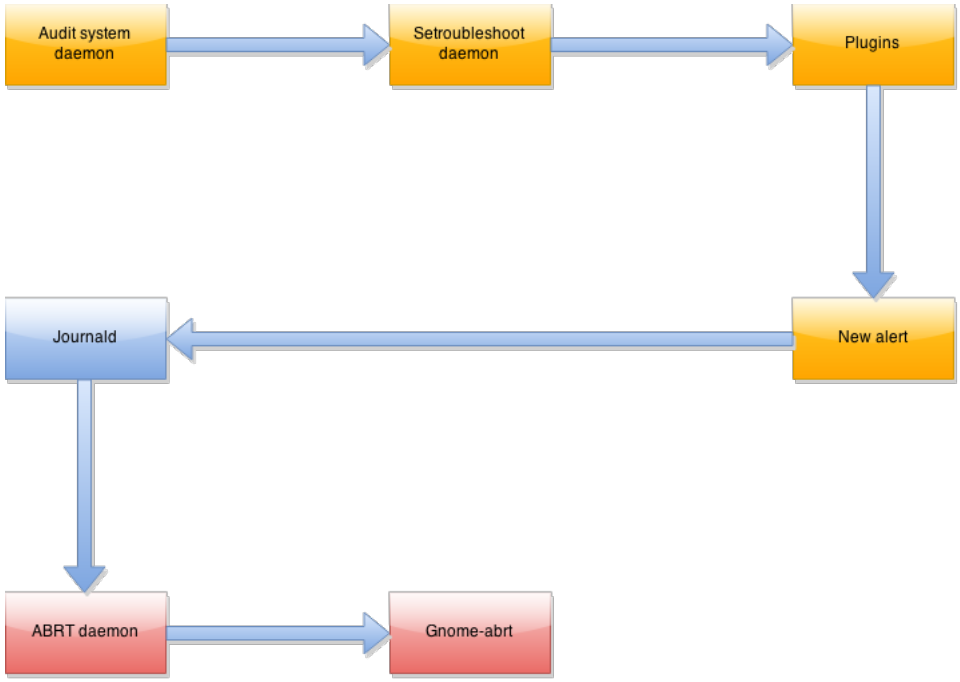
Od predchádzajúceho návrhu sa upustilo a začal sa riešiť podrobnejší návrh, ktorý by dokázal využívať logovací systém **systemd-journald**. Tento logovací systém by bolo možné použiť aj na ukladanie AVC správ, ktoré by bolo možné prezerať aj v textovej podobe, napríklad pri serverových verziách, kde nie je dostupné žiadne uživateľské rozhranie. Dôležitejším významom použitia logovacieho systému by bolo ukladanie samotnej databázi do systé-

mových záznamov. Toto riešenie by bolo vhodné pre všetky funkčné požiadavky a zároveň by bola splnená funkcionalita integrácie.

Po naštudovaní logovacieho systému systemd-journald a konzultovaní návrhov, ako by bolo možné databázu s AVC záznamami uchovávať, bol zvolený nasledujúci návrh.

5.2 Architektúra projektu

Pri navrhovaní architektúry projektu sa dbalo na skutočnosť možného integrovania SELinux záznamov do ďalších nástrojov, určených pre oznamovanie stavu systému uživateľovi v budúcnosti. Preto bolo potrebné navrhnúť taký spôsob ukladania správ, aby bol ľahko dostupný. Z týchto dôvodov je architektúra projektu rozdelená na dve základné časti. Týmito časťami sú Backend a Frontend, pričom backendová časť je navrhnutá tak, aby bolo možné rýchlo a jednoducho implementovať prípadnú ďalšiu integráciu.



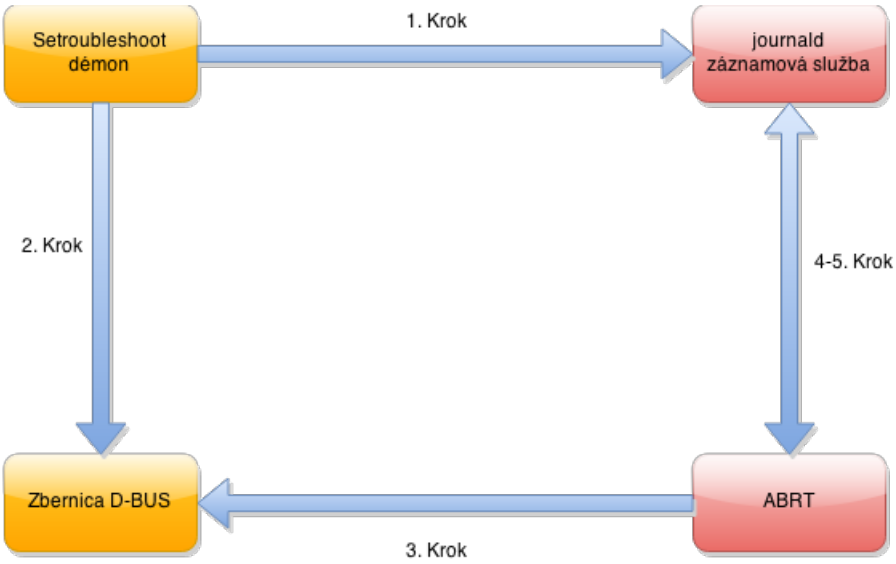
Obrázek 5.1: Nový návrh riešenia

Na Obrázku 5.1 je možné vidieť kompletný tok AVC správy po navrhnutí integrácie spolu s ABRT. Audit démon vygeneruje správu AVC s popisom o odoprení služby, ktorá je odoslaná do setroubleshoot démona pomocou technológie D-BUS. Setroubleshoot démon prevedie sériu analýz daného AVC. Úlohou pluginov je nájsť známe problémy, ktoré sú na strane uživateľa a ako výsledok podať riešenie tejto chyby(napr. zlý kontext objektu). Po túto časť je datový tok AVC správy rovnaký ako existujúce riešenie. Výsledná nová správa je uložená do logovacej služby **systemd-journald**. Táto správa je uložená v prehladnej forme, čitateľnej pre uživateľa. V okamihu keď je správa uložená, setroubleshoot démon, upozorní na túto správu službu ABRT. Táto služba novú správu nájde, a následne spracuje a data uloží do vnútornej ABRT databázy. Na konci je uživateľ upozornený pomocou pro-

gramu **gnome-abrt** formou notifikácie o vzniknutom probléme, ktorý je možné automaticky nahlásiť.

5.2.1 Backend

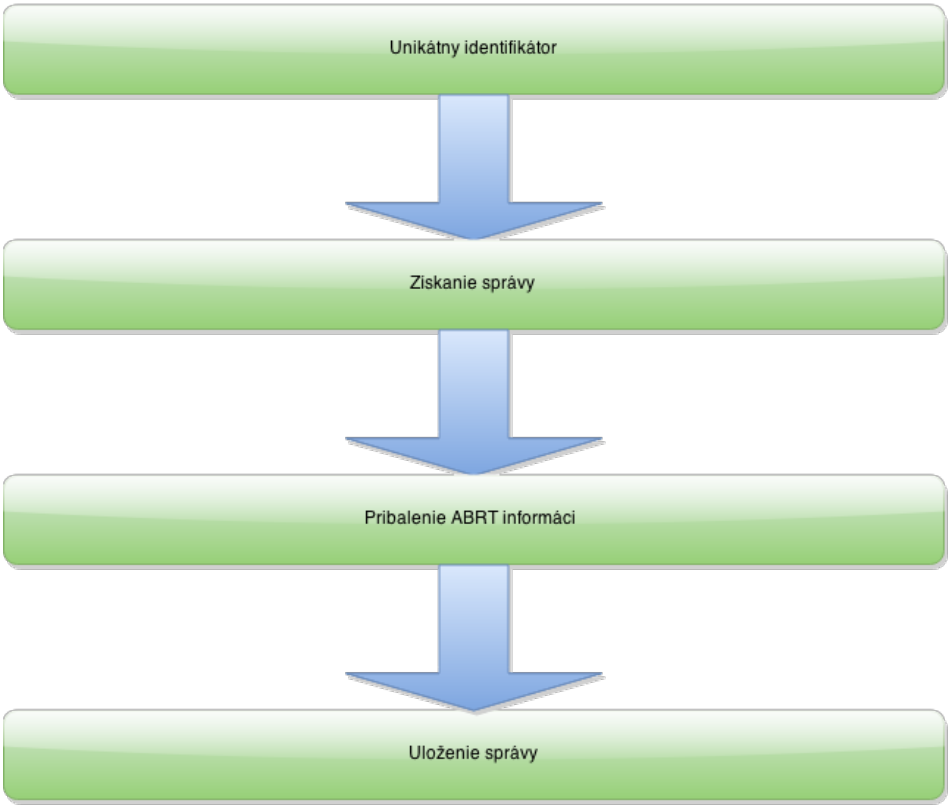
Beckendová časť sa skladá z audit démona pre generovanie správ AVC, démona setroubleshoot pre analýzu, spracovanie a ukladanie AVC správ. Démon setroubleshoot sa stará o celkový dátový tok popísaný vyššie. V tejto službe bude prepracovaná implementácia logovania, kedy namiesto použitia záznamovej služby **syslog** a jeho API, bude použitá novšia záznamová služba **systemd-journald**. Ďalším krokom bude implementácia modulu pre ukladanie prijatých a zanalyzovaných správ do tejto záznamovej služby. Z dôvodu rozdieľnej formy ukladania AVC správy je rovnako potrebné preimplementovať vnútornú datovú reprezentáciu AVC správy, nakoľko doteraz bola správa ukladaná ako reťazec. Do záznamovej služby systemd-journald bude správa ukladaná v reprezentácií podobnej slovníku v programovacom jazyku Python, teda štýlom *kľúč : hodnota*. Spolu s údajmi obsahujúcimi data ku konkrétnej AVC správe sa bude generovať unikátny identifikátor správy, podľa ktorého bude možné AVC správu v záznamovej službe nájsť. Takto spracovaná správa bude uložená do systemd-journald. Do tohto modulu bude implementovaná aj podpora pre technológiu D-BUS, ktorá umožnuje vystaviť signál na jej zbernicu spolu s jedinečným identifikátorom AVC správy uloženej v systemd-journald službe. Po týchto úpravách bude démon setroubleshoot pripravený na integráciu so službou ABRT. Grafický znázornený postup komunikácie je obsiahnutý na obrázku (5.2)



Obrázek 5.2: Backend

Ako už bolo spomenuté, správy typu AVC budú uložené v systemd-journald formou kľúč-hodnota. Tieto údaje ale nebudú bežne viditeľné pre uživateľa z niekoľkých dôvodov. Prvým je množstvo informácií, ktoré samotná správa podáva. Tieto informácie sú často pre uživateľa moc komplexné a obsiahle. Preto bude uživateľ schopný vidieť len základné informácie zo správy AVC a prípadné ponúknuté riešenie problému. Týmto krokom sa zaručí prehľadnosť. Druhým dôvodom použitia takejto datovej štruktúry je fakt, že služba systemd-

journald je schopná uchovávať v štruktúrach, čo umožňuje data uchovávať stále prehladné a ľahko dostupné. Vďaka tejto možnosti nie je potrebné AVC správu ďalej spracovávať v klientskej časti programu ABRT.



Obrázek 5.3: Backend z pohľadu AVC správy

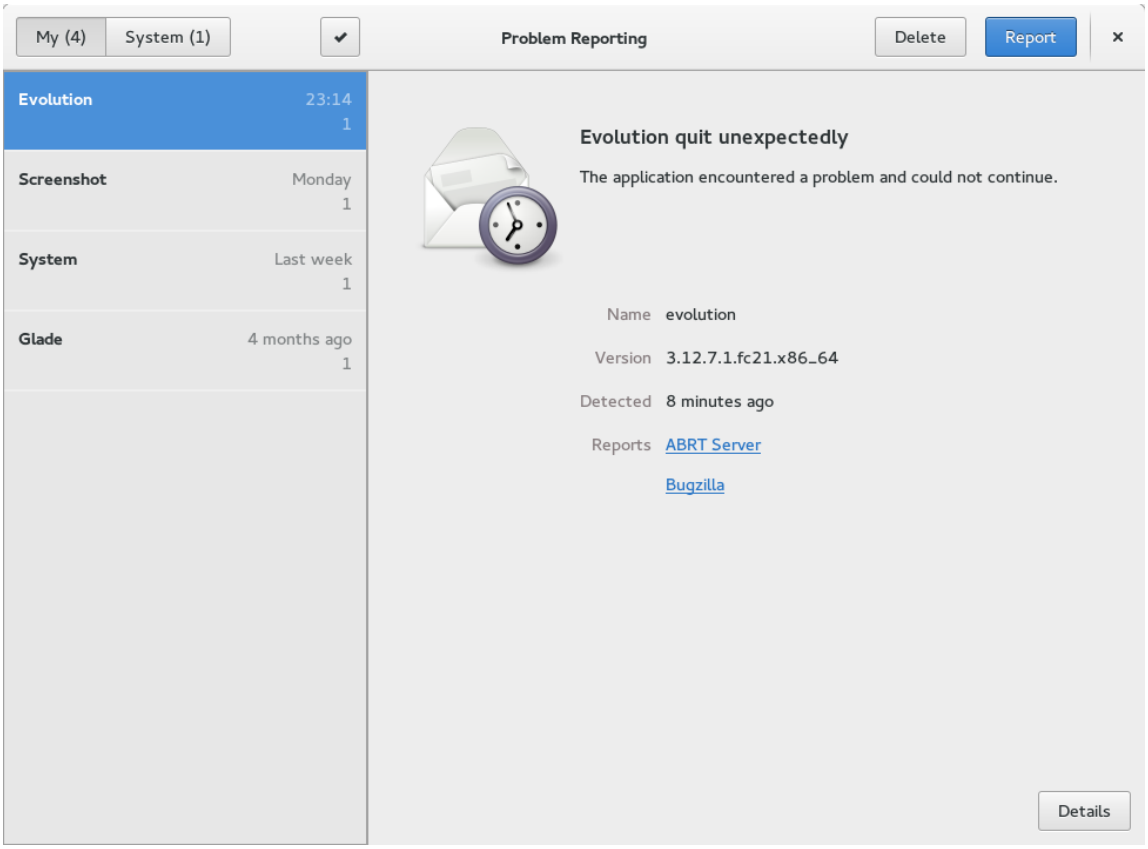
Po analýze a uložení AVC správy je potrebné implementovať plugin do služby ABRT. Podstatou pluginu je vyčkávanie na signál, ktorý vystaví setroubleshoot démon na zbernicu D-BUS. Po detekcii a spracovaní signálu je ABRT službe známy unikátny identifikátor AVC správy, ktorý sa nachádza v záznamovej službe systemd-journald. Nasleduje implementácia časti pluginu démona ABRT, ktorý bude schopný vďaka systémd-journald API nájsť odpovedajúcu správu, v tejto záznamovej službe táto správa bude vystavená službe ABRT a uložená do internej databázy služby, spolu s pribalenými informáciami o tom, ako správu oznámiť uživateľovi a ako ju môže uživateľ nahlásiť(viď. Obr. 5.3).

V tejto časti bol popisaný kompletný návrh implementácie serverovej časti integrácie AVC správ. Takto navrhnutá serverová časť je schopná pracovať nezávisle od klientskej časti. To umožnuje budúcu integráciu s inými klientami pre hlásenie a oznamovanie AVC správ.

5.2.2 Frontend

Pri návrhu integrácie sa dbalo na to aby návrh a implementácia klientskej časti bola čo najednoduchšia a ľahko dosiahnuteľná. Vďaka tomu že backend dokáže ukladať správy do internej databázy ABRT služby, bude možné z časti použiť už existujúce štandardné zobrazovanie správ.

V okamihu, keď je nová správa uložená, je zavolaný program **gnome-abrt**. Tento program slúži na grafické zobrazenie správy v notifikačnej lište. Po kliknutí na ikonu notifikácie je zobrazené okno obsahujúce informácie o chybe, podobné ako na nasledujúcom obrázku(5.4).

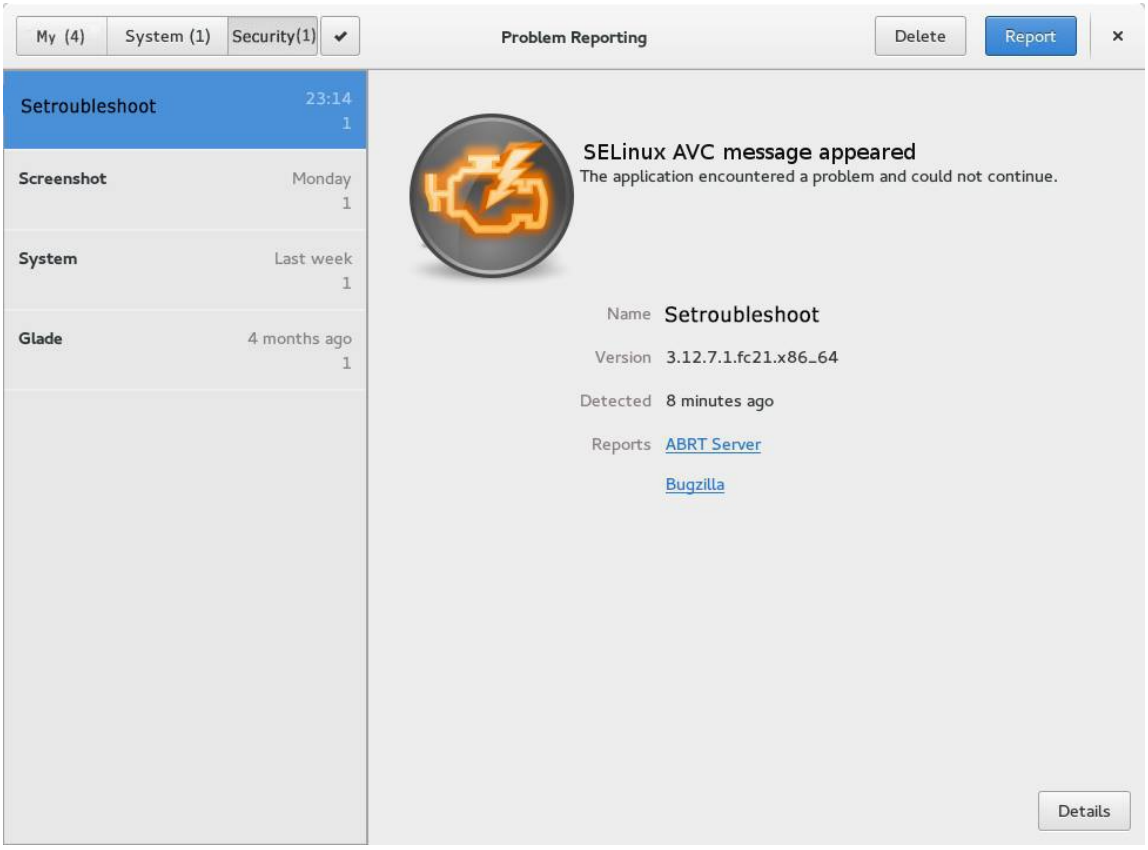


Obrázek 5.4: Frontend

Ako je možné vidiet vyššie, klientský program dokáže zobraziť správy o chybách z databázy, *názov a verziu* programu alebo služby, v ktorej došlo k chybe, kedy došlo k chybe a spôsoby nahlásenia chyby, prípadne je možné túto správu označiť ako prezretú. Pre pokročilých užívateľov je možné zobrazenie aj detailných informácií o chybe. Tam budú zobrazené informácie AVC správy, ktoré sú v záznamovej službe systemd-journald označené ako skryté. Detailné informácie nie sú zobrazované explicitne z dôvodu prehľadnosti, no zároveň sú potrebné v prípade že pokročilejší uživateľ si želá tento chybový stav odstrániť vo vlastnej réžii.

Návrh GUI

Nová grafická reprezentácia AVC správ, bude vychádzať priamo zo zvyšných chybových správ, ktoré ABRT už zobrazuje. Po integrácií bude pridaný ďalší stĺpec v ľavom hornom rohu s názvom Security. Takto bude jednoduché správy filtrovať podľa ich typu. V tomto stĺpci sa budú vyskytovať bezpečnostné chyby, prípadne chyby bezpečnostných mechanizmov(viď. Obr. 5.5).



Obrázek 5.5: Návrh GUI

5.3 Záver návrhu

V tejto kapitole bol popísaný návrh implementácie serverovej časti, ktorá poskytuje zber AVC správ, ich analýzu a ukladanie zanalyzovaných správ do centrálnej záznamovéj služby systému. K tomu bola navrhnutá klientská časť, ktorej úlohou je podanie prehladného popisu bezpečnostnej správy a jej nahlásenia príslušným tímom. Počiatočné návrhy boli diskutované s príslušnými tímami, ktoré by tieto nahlásené správy prijímali. Získal som od nich znalosti, ktoré informácie sú dôležité k analýze problému na strane uživateľa. Podľa toho sa menila štruktúra AVC správy ukladajúcej sa do záznamovej služby. Pri návrhu sa nezamýšlalo nad možnými problémami týkajúcich sa záznamovej služby, ktoré boli objavené v procese implementácie a testovania. Z tohto dôvodu bol návrh niekoľko krát upravovaný. Týmito problémami sa budem zaoberať v ďalšej kapitole.

Kapitola 6

Implementácia

V tejto časti práce bude objasnený postup implementácie, špecifické detaily riešenia, použité technológie a problémy, ktoré sa naskytli v čase implementácie intergrácie. Budú vysvetlené zmeny návrhu a prijaté zlepšenia návrhu, ktoré prišli vo fáze implementácie. Fáza implementácie bola začatá v januári 2015 a jej ukončenie je naplánované na koniec augusta 2015, kedy by integrácia mala byť súčasťou komunitnej distribúcie Fedora a časom aj dstribúcie Red Hat Enterprise Linux. Na konci kapitoly bude popísaný aj aktuálny stav implementácie.

Samotnú implementáciu je možné rozdeliť na 3. časti. Prvá časť začala už vo fo fáze analýzy a návrhu, kedy sa hľadali vhodné technológie použiteľné pre fungujúcu integráciu. Z týchto dôvodov boli programované kúsky kódu, používané na testovanie danej technológie alebo návrhu, pre zistenie či je použitie technologií optimálne alebo vôbec vhodné pre potreby integrácie. Táto časť nepatrí úplne do fázy implementácie ale spomínané kúsky kódu používaného na testovanie boli použité ako odrazový mostík k implementácií, z toho dôvodu je zmienená aj táto časť.

Druhou častou bola implementácia backendu, pričom sa kládol dôraz na prvotné implementovanie správneho ukladania správ do záznamovej služby. Tejto časti bol kladený najvačší dôraz, nakoľko je táto časť kľúčová pre integráciu služby ABRT ale aj ďalších podobných služieb, plánovaných do budúcnosti. Už počas tejto implementácie prišlo k určitým problémom, ktoré bolo nutné riešiť okamžite. Tieto problémy budú spomenuté v ďalších častiach kapitoly. Nasleduje implementovanie pluginu do služby ABRT, ktorá bude AVC správy načítať a následne notifikovať uživateľa o vzniknutom probléme.

Poslednou častou je reimplementácia grafického užívatelského rozhrania a to kvôli podpore AVC správ. V tejto časti bude upravené grafické rozhranie, pridaný stĺpec pre bezpečnostné správy, ako sú napríklad správy AVC.

6.1 Použité technológie

Pre navrhnutý spôsob integrácie AVC správ bolo použitých hneď niekoľko technológií. Pri výbere technológie komunikácie medzi setroubleshoot démonom a ABRT démonom sa prihliadalo k tomu, aby boli tieto technológie aktúalne a používané v distrubúciach Fedora a Red Hat Enterprise Linux. Nasadenie tejto integrácie do komunitnej distribúcie Fedora je naplánované približne na 23. až 24. verziu. Z týchto dôvodov sa používali najnovšie verzie použitých technológií, aby nedošlo k neskorším problémom so zastaralými verziami. Nakoľko ABRT a setroubleshoot sú už existujúce služby, použité technológie boli už zadefinované.

Setroubleshoot démon je naprogramovaný v programovacom jazyku Python. Pridaný

modul pre ukladanie AVC správ je funkčný pre verzie Python 2 aj Python 3. Pre prácu so **systemd-journald** bolo použité API pre jazyk Python. Toto API je rovnako funkčné pre obe verzie jazyka Python.

Pluginy pre služby ABRT sú naprogramované v programovacom jazyku C, preto aj plugin pre načitávanie správ zo záznamovej služby je naprogramovaný v tomto jazyku. Silnou stránkou tohto pluginu je rýchlosť, ktorou sú dáta spracúvané. Zobrazovacia (notifikačná) časť je naprogramovaná opäť v jazyku Python, spolu s použitím grafického toolkitu GTK+.

Už pri návrhu projektu bolo snahou pracovať s modernými technológiami, používanými v najnovších distribúciach Fedora a Red Hat Enterprise Linux, z toho dôvodu bola vybraná technológia D-BUS, ktorá slúži na medziprocesovú komunikáciu. Neskôr bola vybraná aj záznamová služba journald, pochádzajúca z balíka systemd, ktorá slúži na centrálne ukladanie záznamov operačného systému.

Ako nástroj pre vytváranie UML diagramov [12] bol použitý online program *draw.io*, ktorý je integrovaný do cloudového riešenia *Google Drive*. Novo implementované moduly boli programované vo vývojovom prostredí *Kdevelop* [4].

6.2 Postup implementácie

Postup implementácie bol primárne rozdelený na implementáciu spracovania AVC správ, ich ukladanie a ľahké dohľadanie, a následne na implementáciu zvyšných častí integrácie. Tento postup bol zvolený zámerne, nakoľko sa začalo pracovať aj na integrácií AVC správ do iných klientských programov, ako je napríklad *Cockpit*. Z toho dôvodu sa po implementácií ukladania správ prešlo k testovaniu tohto modulu, a až následne bol implementovaný plugin služby ABRT pre prijímanie týchto správ. Tento postup sa ukázal ako osvedčený, nakoľko sa počas testovania ukladania AVC správ objavili rôzne pripomienky a zmeny, vďaka ktorým bol pozmenený aj návrh implementácie pluginu služby ABRT.

6.3 Problémy pri implementácii

Počas implementácie sa objavili problémy, ktoré neboli preberané v čase návrhu implementácie, čo znamenalo značné spomalenie samotnej implementácie. Boli objavené 2 závažné problémy, ktoré budú spomenuté v tejto časti.

Prvým problémom bolo spracovanie duplikátnych AVC správ. V prípade, že audit démon generoval rovnaké AVC správy, pri ktorých bol rozdieľ len čas vzniku udalosti, a tento interval bol veľmi malý (rádovo pár sekúnd), bola záznamová služba zahlcovaná rovnakými správami. Tento stav bol nežiadúci, nakoľko bolo možné veľmi jednoducho zahltiť záznamovú službu, ktorá by mohla následne ostatné správy zahadzovať. Tento problém mohol vážne ohroziť bezpečnosť celého operačného systému, preto sa okamžite začalo pracovať na zmene návrhu, v ktorej by boli duplikátne správy zahadzované už pri prijatí službou setroubleshoot. Problém bol vyriešený tak, že setroubleshoot démon je schopný uchovávať AVC správy, ktoré prijal v nedávnej minulosti a v prípade že práve prijatá správa je duplikátna s niektorou, ktorú práve uchováva, bude novo prijatá správa zahodená. Týmto spôsobom bol vyriešený problém duplikátnych správ a zároveň sa predišlo zahlteniu záznamovej služby.

Druhý problém, ktorý bol objavený počas implementácie, naráža na samotnú implementáciu záznamovej služby a vlastnosti jazyka Python. V prípade, že je audit démon žačne odosielať enormné množstvo AVC správ, ktoré musí setroubleshoot démon spracúvať a následne ukladať do systemd-journald služby, dochádza k spomaleniu tejto služby,

až k možnému zahlteniu. Problémom je pomalé spracúvanie ukladania samotnej správy a obmedzená veľkosť žurnálov tejto služby. Navrhnuté boli dve riešenia problému.

Prvým je potreba zväčšiť veľkosť týchto žurnálov. To však nerieši problém úplne, pretože v niektorých systémoch nie je možné, či už z kapacitných dôvodov alebo iných, zväčšiť veľkosť žurnálov. Druhým riešením bolo vytvorenie fronty v setroubleshoot démonovi, ktorý by tieto správy dokázal podržať a regulovať ich ukladanie na základe veľkosti žurnálov. Tento problém rieši zahltenie žurnálov, no v konečnom dôsledku nezlepšuje rýchlosť doručenia novej správy či už do záznamovej služby alebo do klientskej časti služby ABRT.

6.4 Aktuálny stav

V čase písania tejto časti práce bol naimplementovaný modul setroubleshoot démona, ktorý dokáže spracúvať a ukladať AVC správy do žurnálov a zobrazovať ich v prehladnej forme pri prehladávaní žurnálov. Tento modul obsahuje ošetrenie z predchádzajúcej podkapitoly, ktorá sa zaoberala problémami vzniknutými pri implementácií. Ďalej bol naimplementovaný testovací skript pre testovanie parsovania a ukladania AVC správ v žurnáloch, aktuálne sa pracuje na funkcionalite pluginu pre službu ABRT, ktorá bude tieto správy zobrazovať. V tomto stave existuje funkčný prototyp, ktorý dokáže získať informácie o AVC správe. Na obrázku(6.1) je možné vidieť AVC správu uloženú v záznamovej službe systemd-journald. Je potrebné zapracovať na optimálnom vyhľadávaní a načítaní týchto správ. V poslednom bode implementácie sa bude pracovať na úpravách grafického uživateľského rozhrania. Implementácia je celkovo oneskorená z dôvodu vyskytnutých problémov, no dátum ukončenia implementácie by nemal byť ohrozený.

```
Fri 2015-04-24 22:29:19.299218 CEST
    _BOOT_ID=9a4606e433df4065a790b4ec21d7746f
    _MACHINE_ID=9ac823f0af654d86beea3e30b5744e9b
    _HOSTNAME=Thinkpad
    _TRANSPORT=journal
    _CAP_EFFECTIVE=0
    _GID=1000
    _AUDIT_SESSION=1
    _AUDIT_LOGINUID=1000
    _SYSTEMD_CGROUP=/user.slice/user-1000.slice/session-1.scope
    _SYSTEMD_SESSION=1
    _SYSTEMD_OWNER_UID=1000
    _SYSTEMD_UNIT=session-1.scope
    _SYSTEMD_SLICE=user-1000.slice
    _UID=1000
    SYSLOG_IDENTIFIER=python
    _COMM=python
    _EXE=/usr/bin/python2.7
    _CMDLINE=python journald_AVC.py
    _SELINUX_CONTEXT=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
    ENFORCING_MODE=ENFORCING
    SOURCE=/usr/sbin/NetworkManager
    POLICY_RPM=selinux-policy-3.13.1-125.fc23
    SELINUX_ENABLED=True
    POLICY_TYPE=Targeted
    SCONTEXT=system_u:system_r:NetworkManager_t:s0
    TCLASS=file
    TCONTEXT=system_u:object_r:etc_t:s0
    _PID=8814
    _SOURCE_REALTIME_TIMESTAMP=1429907359299218
```

Obrázek 6.1: AVC správa v systemd-journald

6.5 Záver implementácie

Implementácia funkcionalít začala len nedávno a bola zbrzdená problémami spomenutými vyššie. Z tohto dôvodu nie je v čase písania tejto publikácie dostupná kompletná integrácia v takom stave, ako bola navrhnutá. Napriek tomu je dostupný prototyp na dátovom nosiči, ktorý neobsahuje presné zmeny v grafickom uživateľskom rozhraní. Tento protopyp

je v spustiteľnej verzii a návod na jeho spustenie sa nachádza v prílohe A. Zdrojové kódy budú uvoľnené a bude možné ich nájsť na webovej stránke *www.github.com* po ukončení implementácie. Na implementácií sa bude pracovať ďalej do konca augusta 2015, aby bolo možné túto integráciu doručiť do distribúcie Fedora verzie 23. Kompletná dokumentácia bude dostupná až po implemntácií, momentálne sú dostupné len komentáre k zdrojovým kódom.

Kapitola 7

Záver

Na integrácií sa začalo pracovať v septembri 2014, kedy po komunikácií s ľudmi zo spoločnosti Red Hat bol zadefinovaný projekt, ktorý by bol schopný zaintegrovať bezpečnostné správy AVC do jednotného systému na uchovávanie a nahlasovanie týchto správ. Tento projekt nabral na dôležitosti po stretnutiach s tímom spoločnosti Red Hat, ktorý sa stará o grafické uživateľské rozhranie distribúcií Red Hat Enterprise Linux a Fedora. Ten oznámil, že sa v budúcnosti uvažuje len o službe ABRT, ktorá by tieto chyby nahlasovala a klientská časť služby setroubleshoot, Sealert by bola v neskorších verziach operačných systémov odstránená. Po tomto rozhodnutí som bol nasadený na navrhnutie integrácie, ktorá by integrovala klientskú časť Sealert do nástroja ABRT.

Hlavným prínosom tejto práce bolo nájdenie optimálneho riešenia integrácie a uchovávania AVC správ. Ďalším krokom bola snaha o zjednodušenie procesu nahlasovania AVC správ príslušným týmom v spoločnosti Red Hat. V neposlednom rade bolo cieľom aj zvýšenie prívetivosti bezpečnostného mechanizmu SELinux, ktorý implementuje povinné riadenie prístupu. Od tohto projektu sa očakáva, že bude SELinux technológia viac používaná aj na desktopových systémoch, kde je naozaj potrebná. Príkladom môžu byť aj vážne bezpečnostné zraniteľnosti v operačnom systéme Linux, ktoré prezentuje napríklad kybernetický útok "shellshock", kedy technológia SELinux dokázala minimalizovať možné následky tohto útoku.

V prvých troch kapitolách práce sa venujem teoretickým základom práce. Prehĺbil som svoje poznatky o typy riadenia prístupu a formu implementácie povinného riadenia prístupu. Oboznámil som sa s dátovým tokom AVC správy od jej vygenerovania cez spracovanie až k samotnému upozorneniu uživateľa správou. Neskôr som rozšíril svoje poznatky o formy zaznamenávania správ systému, dôležitosť ich uchovávania a sledovania.

V zvyšných častiach práce sa venujem analýze, návrhu implementácie a samotnej implementácií zadaného projektu. Popisujem aktuálny stav informovania užívateľa o AVC správach a poukazujem na jeho nedostatky. Následne definujem riešenie tohto problému formou návrhu integrácie AVC správ do služby ABRT a zameriavam sa na požiadavky, ktoré boli kladené pri zadávaní projektu. V kapitole implementácie sa zameriavam na problémy, ktoré vznikli pri implementácií a snažim sa poukázať na riešenia týchto problémov. Výsledkom je funkčný prototyp, ktorý čaká posledná fáza testovania a následné nasadenie do vývojárskej verzie distribúcie Fedora.

7.1 Vlastný prínos

Na tomto projekte som pracoval sám, takže som zastával úlohu analyzátora, návrhára a rovnako som integráciu aj implementoval. Po zadaní projektu integrácie som hľadal najoptimálnejšie riešenie, ktoré by spĺňalo všetky body zadania, a teda celý návrh implementácie je moja práca. Na projekte som získal cenné skúsenosti s prácou na open-source projektoch, a svoju prácu som po častiach prezentoval vedeniu firmy Red Hat. Počas analýzy a návrhu som viedol viaceré konzultácie, počas ktorých som sa učil praktikám rôznych tímov. Počas fázy implementácie som obsolvoval odborné konzultácie a diskusie so SELinux tímom, konkrétne s Ing. Miroslavom Greplom o popise konkrétnych funkcionalít démona Setroubleshoot. Rovnako som sním konzultoval vyskytnuté problémy pri implementácií, čím som získal potrebné znalosti na návrh ich riešenia a ich následnú nápravu. Najväčším prínosom bol návrh tejto integrácie, na ktorom som získal skúsenosti a osvojil si navrhovanie vačších projektov v praxi. V neposlednom rade som si prehĺbil znalosti operačného systému Linux.

7.2 Budúcnosť projektu

Budúcnosť projektu sa ukazuje ako pozitívna, projekt zaujal ľudí ako z SELinux tímu tak aj ABRT tímu, s ktorými som mal pravidelné konzultácie. Po dôkladnom testovaní sa moja práca uprie na snahu o zlúčenie tejto integrácie do vývojovej verzie služby ABRT. Takto by bola integrácia súčasťou distribúcie Fedora verzie 23 alebo verzie 24. Nie je vylúčené ani začlenenie tejto integrácie do komerčnej distribúcie Red Hat Enterprise Linux verzie 8.

7.3 Možné rozšírenia

Už počas implemtácie sa objavilo možné rozšírenie tejto integrácie. Jedná sa o vytvorenie pluginu pre službu **Cockpit**, ktorý by dokázal získavať AVC správy rovnakým spôsobom ako nástroj ABRT. Cockpit je služba pracujúca ako webové rozhranie pre správu servera, prípadne väčšieho množstva serverov, ktoré používajú komunitnú distribúciu Fedora. K tomuto rozšíreniu sa pristúpilo a bude sa pracovať aj na tejto integrácií. Po nasadení tejto integrácie a následnej spätnej väzbe od komunity sa budú zvažovať ďalsie kroky k možným rozšíreniam tohto projektu.

Literatura

[1] Arapov, A.; Moskovcak, J.; Prikryl, Z.: Automatic bug reporting tool. Duben 8 2014, uS Patent 8,694,831.

[2] Dixon, R.: *Open source software law*. Artech House, 2004.

[3] Downs, D. D.; Rub, J. R.; Kung, K. C.; aj.: Issues in discretionary access control. In *2012 IEEE Symposium on Security and Privacy*, IEEE Computer Society, 1985, s. 208–208.

[4] Gehrmann, B.; Tennis, C.; Pol, B.: KDevelop User Manual. 2004.

[5] Gerhards, R.: The syslog protocol. 2009.

[6] Grepl, M.: *MAC řízení přístupu*. Diplomová práce, Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, Brno, 2008.

[7] Lindqvist, H.: Mandatory access control. *Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901*, ročník 87, 2006.

[8] Love, R.: Get on the D-BUS. *Linux Journal*, ročník 2005, č. 130, 2005: str. 3.

[9] Mayer, F.; Caplan, D.; MacMillan, K.: *SELinux by example: using security enhanced Linux*. Pearson Education, 2006.

[10] McCarty, B.: *SELinux*. Reilly Media Inc, 2004, iSBN 0-596-00716-7.

[11] Morris, J.: Secure and simple sandboxing in selinux.

[12] Oestereich, B.: *Developing Software with UML: Object-oriented analysis and design in practice*. Pearson Education, 2002.

[13] Peter Loscocco, N.: Integrating flexible support for security policies into the Linux operating system. In *Proceedings of the FREENIX Track:... USENIX Annual Technical Conference*, The Association, 2001, str. 29.

[14] Samarati, P.; de Vimercati, S. C.: Access control: Policies, models, and mechanisms. In *Foundations of Security Analysis and Design*, Springer, 2001, s. 137–196.

[15] Schwarz, M. A.: Take Command: The System Logging Daemons, syslogd and klog. *Linux Journal*, ročník 2000, č. 75es, 2000: str. 16.

[16] Summerfield, M.: *Programming in Python 3: a complete introduction to the Python language*. Addison-Wesley Professional, 2010.

[17] Vermeulen, S.: *SELinux System Administration*. Packt Publishing Ltd, 2013.

[18] Vermeulen, S.: *SELinux Cookbook*. Packt Publishing Ltd, 2014.

[19] Walsh, D. J.; Engineer, S. L.; MacMillan, K.: Managing red hat enterprise linux 5. In *SELinux Symposium*, 2007.

[20] Welsh, M.; Dalheimer, M. K.; Kaufman, L.: *Running Linux*. O'Reilly & Associates, Inc., 1999.

[21] Wright, C.; Cowan, C.; Smalley, S.; aj.: Linux security modules: General security support for the linux kernel. In *Foundations of Intrusion Tolerant Systems*, IEEE Computer Society, 2003, s. 213–213.

Příloha A

Návod na generovanie AVC správy

Nasledujúci text obsahuje návod na nainštalovanie služby ABRT a setroubleshoot démona, ktoré podporujú vzájomnú integráciu. Všetky potrebné zdrojové kódy sú dostupné na dátovom nosiči priloženom k tejto práci.

Návod:
1. krok: Nainštalovajte komunitnú distribúciu Fedora 21.
2. krok: Nakopírujte všetky balíčky a skript z dátového nosiča do jednej složky.
(napr. */tmp/bp/*)
3. krok: Nainštalujte balíčky príkazom: `# yum install ./(názov balíka)`

Momentálne sú nainštalované všetky potrebné balíky a ich závislosti. Následne je možné vygenerovať AVC správu. Táto správa bude vygenerovaná okamžite pri spustení nasledujúceho skriptu. Skript úmyselne zmení kontexty logovacích súborov služby mariadb, ktorá nebude mať právo čítať tieto súbory. Takto dôjde k odopretiu tejto akcie.
Spustenie skriptu: `# ./generateAVC.sh`

Okamžite po spustení bude uživateľ infomovaný službou ABRT o výskyte novej AVC správy a riešení ako tento problém vyriešiť. V prípade že nástroj ABRT neponúkne riešenie, je možné túto správu nahlásiť. Tento krok je možné vykonať stlačením tlačidla "report". Následne je potrebné vyplniť potrebné kontaktné údaje a správa je úspešne nahlásená.