Instituto Federal de Educação, Ciência e Tecnologia de Brasília, campus Taguatinga,
Campus Taguatinga

COMPARAÇÃO DE FERRAMENTAS DE SANDBOXING EM SISTEMAS GNU/LINUX

Por

ELLIAN ARAGÃO DIAS, JOÃO VITOR SOUZA REZENDE

Trabalho de Graduação

BRASÍLIA/2023

Ellian Aragão Dias, João Vitor Souza Rezende

COMPARAÇÃO DE FERRAMENTAS DE SANDBOXING EM SISTEMAS GNU/LINUX

Trabalho apresentado ao Curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia de Brasília, campus Taguatinga, como requisito parcial para obtenção do grau de Bacharel em Bacharelado em Ciência da Computação.

Orientador: Daniel Saad

BRASÍLIA
2023

Ellian Aragão Dias, João Vitor Souza Rezende
Comparação de ferramentas de sandboxing em sistemas GNU/Linux/ Ellian Aragão Dias, João Vitor Souza Rezende. – BRASÍLIA, 2023-
45 p. : il. (algumas color.) ; 30 cm.

Orientador Daniel Saad

Trabalho de Graduação – Instituto Federal de Educação, Ciência e Tecnologia de Brasília, *campus* Taguatinga,, 2023.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 004

Trabalho de Graduação apresentado por **Ellian Aragão Dias, João Vitor Souza Rezende** ao curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia de Brasília, *campus* Taguatinga, sob o título **Comparação de ferramentas de sandboxing em sistemas GNU/Linux**, orientado pelo **Prof. Daniel Saad** e aprovado pela banca examinadora formada pelos professores:

Prof. Fabiano Cavalcanti Fernandes
Computação/IFB

Prof. Leandro Vaguetti
Computação/IFB

Prof. Roberto Duarte Fontes
Computação/IFB

Prof. Raimundo Cláudio da Silva Vasconcelos
Computação/IFB

Prof. Daniel Saad Nogueira Nunes
Computação/IFB

BRASÍLIA
2023

Obrigado a todos que acreditaram que eu poderia ser um belo garoto de programa

Agradecimentos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

When one finds a hard problem, the more complicated it is, the more one ought to work towards enlightening it's solution.

—MÁRCIO DE DEUS

Resumo

add intro

Palavras-chave: add keyword

Abstract

For the implementation of the proposed architecture, we developed a strategy to automate **cr!** (**cr!**) assignments which is based on two main components: a **rbes!** (**rbes!**) and an **ir!** (**ir!**) model. The strategy coordinately applies these two components in different steps to find the potential developer to a **cr!**. The **rbes!** takes care of the simple and complex rules necessary to consider contextual information in the assignments, e.g., to prevent assigning a **cr!** to a busy or unavailable developer. Since these rules vary from one organization/project to another, the **rbes!** facilitates their modification for different contexts. On the other hand, the **ir!** model is useful to make use of the historical information of **cr!** assignments to match **cr!**s and developers.

Keywords: add keyword

Sumário

1

Introdução

justificar a existência do trabalho e porque ler

I have yet to see any problem, however complicated, which, when looked at in the right way, did not become still more complicated.

—POUL ANDERSON

Software maintenance starts as early as the first software artifacts are delivered, and is characterized by its high cost and slow speed of implementation (**?**). It has been stated that it is the most expensive activity of software development, taking up to 90% of the total costs (**??**). However, despite of the high cost, it is mandatory to ensure the success of the software project. **?** argues, in his *Continuing Change* law of software evolution, that the modification of software is a fact of life for software systems if they are intended to remain useful. **?** reinforced such an argument for the specific case of useful and successful software, where almost all of them have a common practice of stimulating user-generated **cr!** (**cr!**). Actually, software maintenance is driven by **cr!**s reported by many stakeholders, such as developers, testers, team leaders, managers, and clients.

In this context, the **cr!** repositories play an important role in the maintenance and evolution process, being actually a focal point of communication and coordination for software projects (**?**). Through a **cr!** repository, the developers manage and coordinate the corrections and new features to be implemented in the software under development or maintenance. Moreover, the data stored in such repositories are a valuable source of information about the project, which can be used to assist in cost estimation, impact analysis, traceability, planning, expertise discovery, and software understanding (**?**). Examples of these repositories are Mantis (**?**), Bugzilla (**?**), and Trac (**?**).

Briefly, a **cr!** describes a defect to be fixed, an adaptive or perfective change, or a new functionality to be implemented in a software system (**?**). Each **cr!** stores a variety of fields of free text and custom fields defined according to the necessity of each project. In Trac, for example, it has fields for summary and detailed description of a **cr!**. In the same **cr!**, it can be also recorded information about software version, dependencies with other **cr!**s (such as **cr!**s

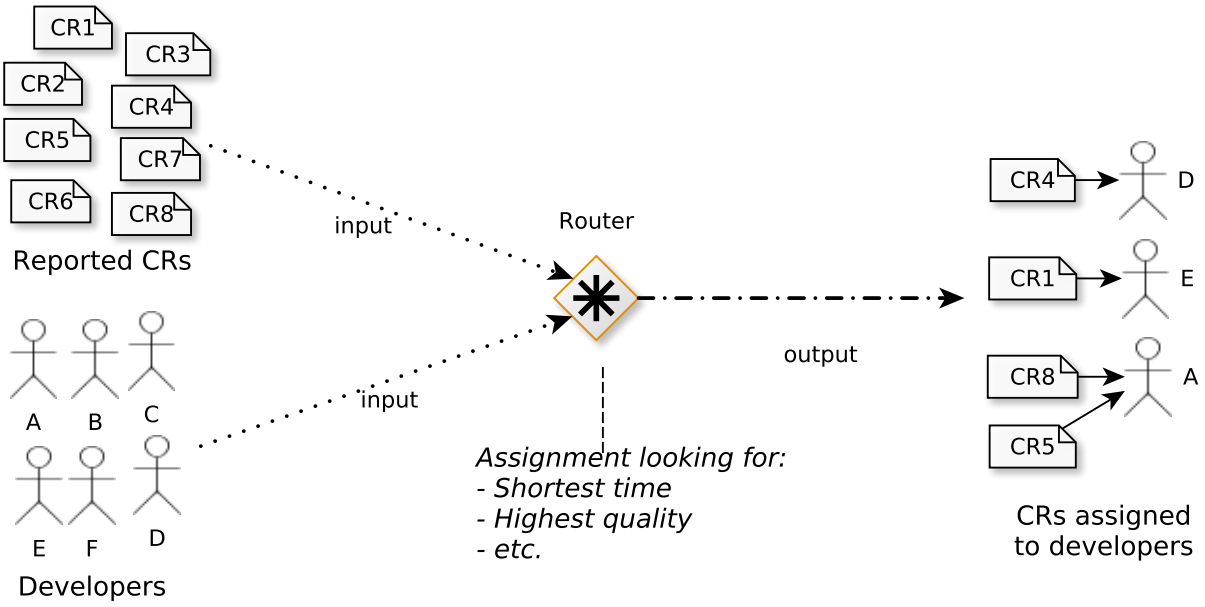
that are blocked, similar, or duplicate), the person who will be assigned to the **cr!**, among other relevant information. Moreover, during the life cycle of a **cr!**, different kinds of discussion take place through the comments that are inserted in it, such as fixing alternatives, workarounds, and architectural decisions (**?**).

1.1 Motivation (Why to Automate cr! Assignment?)

Despite **cr!** repositories claimed benefits to software maintenance and evolution, handling **cr!**s is not cost-free. For example, when new **cr!**s are reported they must be assigned to developers which have adequate expertise to address the request (**???**). Finding the appropriate developer is crucial for obtaining the lowest, economically feasible, fixing time (**?**). Nevertheless, assigning **cr!**s to developers is a labor-intensive and time consuming task (**??**). Indeed, depending on the project being developed, the amount of **cr!**s that are reported and need to be assigned can vary from dozens to hundreds per day (**?**).

Figura 1.1 shows the activity of assigning **cr!**s. At the top-left corner of the figure, there are the **cr!**s which have been reported to the software project. At the bottom-left corner, there is a set of developers which could be assigned to those **cr!**s. Then, at the center, the assignment of **cr!**s is performed; the **cr!**s and developers must be matched, and each developer should fix one or more **cr!**s. Commonly, the matching is performed aiming at the shortest time and the highest quality for the **cr!** fixing activities.

Figura 1.1: cr! assignment. The router, which may be the **ccb!**, project leaders, or managers, must match **cr!**s and developers in order to obtain the shortest fixing and highest quality.



Source: Made by the author.

Nevertheless, by increasing the amount of reported **cr!**s or the size of the development team, it is visible that the router becomes overloaded and the **cr!** assignment becomes an intensive, error prone activity. It was confirmed by **?**, which identified that 37%-44% of the **cr!**s in Mozilla

and Eclipse projects did not reach the right developer in the first assignment. These **cr!**s, in turn, had their fixing time delayed because they needed to be reassigned one or more times. Furthermore, if the **cr!**s are not fixed by the appropriate developers, there is also the chance of introducing new defects during the **cr!**s fixing.

In this context, we believe that it is necessary to develop methods and tools to automate the assignment of **cr!**s and ensure that the **cr!**s are being assigned to the appropriate developers. With these methods and tools, we could reduce the time needed to perform the assignments and, given that the appropriate developers are actually being selected, the quality and time for the **cr!** fixing are also improved.

1.2 Problem Statement

As previously mentioned, software maintenance has been considered as the most costly aspect of software development (**?**). There is a myriad of reasons for this situation. One of them is the many changes that are required after software delivery due to poor documented and misunderstood requirements, or simply because *"the clients do not know what they want"* (**?**).

Another reason is the fact that a set of development activities must be inevitably performed in order to implement a change. For instance, for each change to be implemented it is necessary to comprehend the existing software artifacts, modify the software's source code to implement the change, perform tests and verification, and deliver the new version of the software. Additionally, very often, the implementation of the change ends up by introducing new defects in the software.

A third reason is the management aspects of software maintenance. It is necessary to keep track of all these changes that are performed, generally considering different versions of the software and customers.

1. Firstly, the approaches available in the literature were designed to perform autonomously. That is, the software analysts do not have the control of the approach; they cannot modify the approach's behavior. Without such control, in turn, the approach cannot be properly calibrated. As a consequence, if the approach's performance is not satisfactory, it is simply discarded.

2. Secondly, the reported values for accuracy of these approaches are still low. With low accuracy, the previous reason takes place. That is, as the approaches perform with low accuracy, and the software analysts do not have control over them, the approaches are simply discarded.

3. Finally, the third reason concerns the lack of contextual information in those approaches. As is well known, software development companies are dynamic: developers move from projects; developers are hired/fired; developers enter in vacation or take a day off; and developers have different experiences. This dynamic influences

the assignment of **cr!**s. Thus, contextual information is a necessity in automated approaches.

Based on this context, the main research question investigated by this thesis is:

Research question *Is it possible to develop a new approach for automated **cr!** assignment with satisfactory accuracy, leveraging contextual information, and designed in order to put the software analysts in control of such approach?*

With the objective to answer this question, it is necessary to understand current approaches available in the literature, choose the correct technologies that could support dynamic environments and, mainly, understand the necessities of software analysts regarding a new approach for automated **cr!** assignment. Thus, the goal of the work described in this thesis can be stated as:

Research objective *This work proposes an automated approach for **cr!** assignment which uses **ir!** (**ir!**) models, expert systems, and context-aware information in order to select the appropriate developers. The approach is supported by the state-of-the-art in the management of **cr!**s as well as by the understanding of the aspects concerning the **cr!** assignment activity itself.*

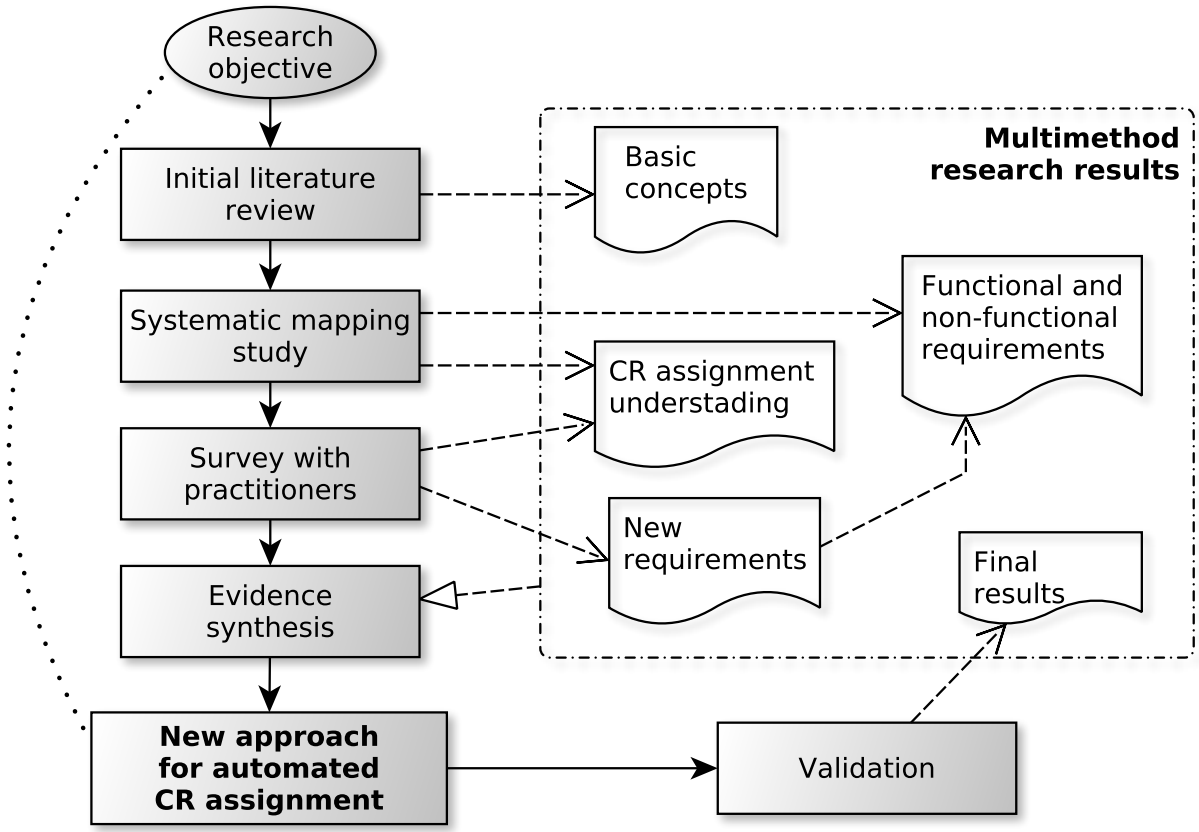
1.3 Research Methodology

This research design of this thesis is based on a multimethod approach (**?**). Such approach combines two or more quantitative (or qualitative) methods in a single study, such as a survey and an experiment (**?**). Multimethod must not be confused with mixed method. In this last, methods for both qualitative and quantitative types of research are applied in a single study. On the other hand, multimethod studies combine different methods for a single research type.

When applying a multimethod approach, the triangulation is used to consolidate the results from the different methods, considering, however, that the same research question(s) was/were investigated in these methods. As a consequence, the triangulation of methods enhances the conclusions and completeness of the study, bringing more credibility to the research findings (**?**). Figura 1.2 shows the multimethod research design applied in this thesis.

The design started by stating the research objective, which we defined in Seção 1.2, and performing the initial literature review. This last provided the basic concepts and understanding of the area. Then, a systematic mapping study and a questionnaire-based survey were conducted. These two gathered detailed information on our research topic. Indeed, both of them were used to understand the key aspects of **cr!** assignment and identify the set of requirements to automate the assignments. In the evidence synthesis step, these results were detailed and organized in order to formulate the approach to automate **cr!** assignments, which was constructed in the next step. Finally, the research design states the validation of the proposed approach.

Figura 1.2: The research methodology applied for this thesis.



Source: Made by the author.

1.4 Out of Scope

As the proposed approach is part of a broader context, a set of related aspects will be left out of its scope. Thus, the following topics are not directly addressed in this thesis:

1. **Tools for cr! management.** We are addressing a specific aspect of **cr!** management, which is the **cr!** assignment activity. Thus, it is out of scope of this thesis to provide a complete solution for **cr!** management. Instead, we are planning to implement standalone software which will be able to integrate with the most well known tools for **cr!** management, such as Mantis, Bugzilla, and Trac, providing a service to leverage the automation of **cr!** assignments.

2. **Software maintenance process.** Software maintenance involves a set of activities aiming at implementing modifications in some software project. These activities must be coordinated through a process so that the maintenance can be successful. In Chapter 2, we discuss some of these processes. However, in this thesis, we are not concerned with the maintenance process itself. Actually, it should be transparent in our approach to automate **cr!** assignment. Thus, it is out of scope of this thesis to provide any process assessment for software maintenance beyond the activity of **cr!** assignment.

3. **ir! models.** Many models for **ir!** have been proposed for different objectives, including the **cr!** assignment itself. However, due to the broad availability of these models, it is out of scope of this thesis to develop a new one. Instead, the **ir!** models with better performance, identified through the systematic mapping study, were chose to be integrated in our approach;

4. **Rule-based expert systems.** Similar to **ir!** models, rule-based expert systems have a long history of development. Thus, our approach does not intend to develop a whole new system with this purpose. Actually, we integrated in our approach the Drools[1] expert system, which is a mature tool that can be easily manipulated;

5. **Mathematical formulations on NP-Complete problems.** We understand that the problem of assigning **cr!**s to software developers is in the broad category of *assignment problems*, which is well known to be NP-Complete. Thus, could be formulated as such. However, the mathematical formulations of the **cr!** assignment problem is out of scope of this thesis. As well as finding an optimal solution on the context of NP-Complete problems is also out of scope. The main reason for this is the human factors and context variables that are involved in the assignment of **cr!**s, which make this problem hard to be computable. A mathematical formulation of the **cr!** assignment problem is provided by **?**.

1. An overview of the software maintenance concepts and processes, with emphasis on the importance of **cr!** management aspects;

2. A survey performed with practitioners from a large organization, in order to understand the aspects of the **cr!** assignment activity. Published in the *17th International Conference on Evaluation and Assessment in Software Engineering (EASE'2013)* (**?**);

3. A replication of the previous survey in two more organizations;

4. A systematic mapping study performed to understand the challenges and opportunities of **cr!** management, as well as to identify research gaps and the road ahead. Accepted for publication in the *Journal of Software: Evolution and Process* (**?**);

5. The definition of the functional and non-functional requirements that are required to effectively automate **cr!** assignment, which takes as input the systematic mapping study and the survey;

6. The definition of an approach that satisfies the identified requirements to automate the **cr!** assignment activity;

[1] `http://www.jboss.org/drools/`

7. The realization of the proposed approach's architecture, in which we described the methods and techniques used for the implementation, as well as the components that have to be built and the third party components that should be assembled together in order to provide a service for automated **cr!** assignment; and

8. The evaluation of the proposed approach, performed as an offline experiment simulating a real context.

2

Referencial teórico

Como base do conteúdo a ser referenciado dentro deste trabalho, existem diversas tecnologias e conceitos que são de fundamental importância a compreensão de modo que seja entendido desde os comparativos a de fato como as ferramentas a serem tratadas funcionam e interagem com o kernel Linux. Diante disto, temos a seguir alguns conceitos a serem expostos de modo a clarificar a sua importância e a base das ferramentas a serem tratadas, nestas temos namespaces e cgroups como os conceitos mais importantes e utilizados diante de todos os softwares a serem comentados posteriormente.

2.1 Namespace

A tecnologia consiste em fazer o isolamento dos recursos a serem utilizados por determinado software que esteja rodando. Deste modo o namespace é uma camada de interação que se encontra entre a sua execução propriamente dita e sua interação com kernel, tal como um proxy SCHEEPERS (2020).

Como comentado, esta camada que encontra-se entre o processo e o kernel, atua quando determinado software executa uma syscall (chamada de sistema) e as estruturas do processo, possuem os dados de determinado namespace, este que por sua vez delimita que recursos estão disponíveis para serem utilizados.

Desta forma cria-se uma virtualização dos recursos que são disponíveis e quais são eles, uma vez que a resposta para o software a ser executada é dada pelas configurações daquele namespace sob o qual a aplicação roda. Sendo esta a base para criação de containers, dos quais trata-se de um processo o qual roda de maneira isolada sob um sistema operacional.

Por fim, uma vez compreendida sua função, é necessário a compreensão de quais tipos de namespaces estão disponíveis uma vez que utilize-se o kernel posterior 5.6 do linux.

2.2 Cgroups

O controle de grupos, também chamado de cgroups, é a ferramenta para controle de uso do recurso, sendo esses de alguns tipos diferentes, mas em seu cerne trata-se de controle de

quantidade de uso deste, ou seja quanto tempo ou banda de utilização de determinado recurso estaria liberado para um determinado processo do qual se encontra sob aquele cgroup.

Compreendido que se trata do uso de recurso, aqueles que estão sob controle de determinado grupo, pode-se limitar o uso da CPU e utilização de blocos de I/O (entrada e saída de dados). Deste modo pode-se determinar quanto tempo de processamento está disponível e/ou quantos bytes estão disponíveis na entrada e saída do processo, seja ela utilização de rede (internet), escrita e leitura de arquivos, etc...

2.3 Containers

2.4 Virtualização

3

Ferramentas de Sandboxing

3.1 AppArmor

O AppArmor é um sistema de controle de acesso obrigatório (MAC - Mandatory Access Control) que oferece recursos de segurança adicionais ao sistema operacional Linux. Ele permite aos administradores do sistema criar políticas de segurança personalizadas para controlar o acesso de processos e aplicativos a recursos do sistema, como arquivos, diretórios, portas de rede e outros.

O objetivo do AppArmor é restringir o acesso a esses recursos, limitando as ações que um processo pode realizar, a fim de evitar a execução de código malicioso ou não autorizado. Ao utilizar o AppArmor, os administradores podem criar perfis de segurança que definem as permissões e restrições para cada aplicativo em execução no sistema.

3.1.1 Configuração

A configuração do AppArmor envolve a definição de perfis de segurança para cada aplicativo. Esses perfis especificam quais recursos do sistema o aplicativo pode acessar e quais ações ele pode realizar. Essa configuração pode ser feita por meio de arquivos de perfil ou usando ferramentas de linha de comando fornecidas pelo AppArmor.

Embora o AppArmor seja uma ferramenta poderosa para aumentar a segurança do sistema, é importante notar que ele não é uma solução completa. É recomendado que os administradores de sistema adotem uma abordagem em camadas para a segurança, combinando o uso do AppArmor com outras medidas de proteção, como atualizações regulares de segurança e práticas de segurança recomendadas.

3.1.2 Integração com linux

Uma das principais características do AppArmor é sua integração com o kernel do Linux. Ele aproveita os recursos de segurança fornecidos pelo kernel, como namespaces e cgroups, para isolar os processos em seus próprios ambientes protegidos. Essa abordagem garante um bom desempenho e uma pegada de recursos mínima.

3.1.3 gerenciando os perfis

O AppArmor, no Linux, oferece um sistema flexível de gerenciamento de perfis que permite aos administradores do sistema controlar as políticas de segurança de forma granular. Os perfis são arquivos de configuração que definem as permissões e restrições de acesso para cada aplicativo ou processo em execução no sistema.

No AppArmor, o gerenciamento de perfis envolve três etapas principais: criação, configuração e aplicação dos perfis.

A criação de um perfil é a etapa em que um administrador define as permissões e restrições desejadas para um aplicativo específico. O perfil é geralmente escrito em uma linguagem de configuração, como o AppArmor Profile Language (AAPL) ou usando uma abordagem baseada em aprendizado de perfil, onde o AppArmor registra as ações do aplicativo durante uma sessão de treinamento e cria um perfil com base nesses registros.

A configuração de um perfil envolve especificar as permissões necessárias para que o aplicativo funcione corretamente. Isso pode incluir permissões para acessar arquivos, diretórios, dispositivos, sockets de rede e outras entidades do sistema. As restrições também podem ser aplicadas, limitando as ações que o aplicativo pode executar, como impedir o acesso a certas partes do sistema de arquivos ou restringir o uso de recursos de rede.

Uma vez que o perfil tenha sido criado e configurado, ele pode ser aplicado ao aplicativo ou processo específico. O AppArmor suporta diferentes métodos de aplicação de perfil, como associar o perfil a um binário específico, atribuí-lo a um namespace ou usar as ferramentas de linha de comando do AppArmor para aplicar o perfil dinamicamente.

Além disso, o AppArmor suporta o conceito de herança de perfil, onde é possível criar perfis base e perfis dependentes. Os perfis dependentes herdam as permissões e restrições dos perfis base, permitindo a criação de hierarquias de perfis que facilitam o gerenciamento de políticas de segurança em larga escala CONTRIBUTORS (2021).

É importante destacar que o AppArmor fornece recursos avançados de auditoria e registro de eventos. Os logs do AppArmor registram violações de segurança, permitindo aos administradores analisar e solucionar problemas relacionados à configuração de perfil e detectar possíveis ameaças.

3.1.4 exemplos

Considere um servidor web que executa um aplicativo da web. Podemos criar um perfil de segurança específico para esse aplicativo, limitando seu acesso apenas aos arquivos e diretórios necessários para operar corretamente. Isso restringe o impacto de um possível ataque, limitando as ações que um invasor pode realizar dentro do ambiente restrito do aplicativo.

Outro exemplo é o uso do AppArmor em ambientes de desktop, onde aplicativos como navegadores da web são comumente usados. Ao aplicar perfis de segurança a esses aplicativos, podemos restringir seu acesso a informações confidenciais, como arquivos pessoais do usuário,

protegendo assim a privacidade e evitando a exfiltração de dados.

3.2 Bubblewrap

Ferramenta de sandboxing amplamente utilizada no ecossistema Linux. Ele fornece um ambiente seguro e isolado para a execução de aplicativos, permitindo que eles operem com um nível mínimo de privilégios e restrições de acesso. O Bubblewrap baseia-se em recursos avançados do kernel do Linux, como namespaces e cgroups, para oferecer isolamento de processos e limitação de acesso.

3.2.0.1 Isolamento de processos

O isolamento de processos é uma das principais funcionalidades do Bubblewrap. Ele cria um ambiente separado para a execução de aplicativos, isolando-os uns dos outros e do sistema operacional hospedeiro. Cada aplicativo em execução dentro do sandbox do Bubblewrap é encapsulado em seu próprio conjunto de namespaces, incluindo namespaces de processo, rede, sistema de arquivos e hostname. Isso impede que os processos dentro do sandbox afetem outros processos ou acessem recursos do sistema não autorizados.

3.2.1 Limitação de acesso

A limitação de acesso é outra característica fundamental do Bubblewrap. Ele permite que os administradores configurem políticas de segurança detalhadas para restringir o acesso dos aplicativos a recursos específicos do sistema. Por exemplo, é possível limitar o acesso a arquivos, diretórios, dispositivos, sockets de rede e outros recursos essenciais. Essa abordagem granular ajuda a evitar que aplicativos comprometidos acessem dados sensíveis ou executem operações indesejadas.

Sendo amplamente utilizado em diferentes cenários. É uma parte essencial do ecossistema de empacotamento de aplicativos Flatpak, permitindo que os aplicativos sejam executados em um ambiente seguro e isolado. É bastante utilizado em outras tecnologias de empacotamento, como o Snap, para garantir a segurança e o isolamento de aplicativos. Ele também pode ser usado para criar ambientes de teste isolados, facilitando a execução de aplicativos sem risco de impacto no sistema operacional hospedeiro.

3.2.2 Exemplo

Um exemplo prático do uso do Bubblewrap é a execução de um navegador da web em um sandbox. Ao executar o navegador dentro de um sandbox do Bubblewrap, é possível limitar seu acesso ao sistema de arquivos, restringir permissões de rede e impedir que ele acesse informações confidenciais. Isso aumenta a segurança, reduzindo a superfície de ataque em caso de exploração de vulnerabilidades do navegador.

3.3 Capsicum

Ferramenta de segurança e sandboxing desenvolvida para o sistema operacional FreeBSD, embora também tenha sido portado para outros sistemas, como o Linux. Ele oferece recursos avançados de isolamento de processos e restrição de acesso, com foco na minimização de vulnerabilidades e na proteção de aplicativos contra ataques cibernéticos.

As principais características do Capsicum é a implementação do modelo de sandbox baseado em recursos, que permite aos desenvolvedores restringir o acesso de aplicativos somente aos recursos necessários para sua execução. Isso é alcançado através do conceito de "capabilidades"(capabilities), que são permissões granulares que um processo pode ter para acessar recursos do sistema. O Capsicum permite que os desenvolvedores reduzam essas capabilidades para um processo, limitando assim seu poder e evitando a exploração de vulnerabilidades.

3.3.1 Separação de contexto

Tendo a separação de contexto, que é alcançada através do uso de descritores de arquivos e recursos encapsulados chamados "sandbox". Essa separação de contexto permite que um processo tenha acesso somente a um subconjunto específico de recursos, limitando sua capacidade de interagir com outros processos ou modificar recursos não autorizados.

Suporta o uso de descritores de arquivos com limitação de escopo, onde um processo pode restringir o acesso a um conjunto específico de descritores de arquivos. Isso ajuda a prevenir a escalada de privilégios, impedindo que um processo acesse outros arquivos ou recursos além dos permitidos.

3.3.2 exemplos

Uma aplicação prática do Capsicum é a proteção de aplicativos com vários componentes que precisam interagir entre si, mas sem expor todos os recursos do sistema a cada componente. Com o Capsicum, é possível definir capabilidades específicas para cada componente, limitando assim seu acesso a recursos específicos e reduzindo o impacto caso algum componente seja comprometido.

3.4 Docker

3.5 Firejail

3.6 LXC

3.7 OpenVZ

3.8 Podman

3.9 Seccomp

4

Metodologia

5

Analise Comparativa

6

Conclusion

6.1 Introduction

6.2 Section

6.2.1 Subsection

Referências

CONTRIBUTORS, D. H. **AppArmor**. [Acessado em 24 de abril de 2023], `https://debian-handbook.info/browse/pt-BR/stable/sect.apparmor.html`.

SCHEEPERS, C. **Linux Containers and Virtualization**: a kernel perspective. Berkeley, CA: Apress, 2020.

Apêndice

A

Mapping Study's Instruments

Tabela A.1: List of conferences on which the searches were performed.

Acronym	Conference
APSEC	Asia Pacific Software Engineering Conference
ASE	IEEE/ACM International Conference on Automated Software Engineering
CSMR	European Conference on Software Maintenance and Reengineering
ESEC	European Software Engineering Conference
ESEM	International Symposium on Empirical Software Management and Measurement
ICSE	International Conference on Software Engineering
ICSM	International Conference on Software Maintenance
ICST	International Conference on Software Testing
InfoVis	IEEE Information Visualization Conference
KDD	ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
MSR	Working Conference on Mining Software Repositories
OOPSLA	Object-Oriented Programming, Systems, Languages and Applications
QSIC	International Conference On Quality Software
SAC	ACM Symposium on Applied Computing
SEAA	EUROMICRO Conference on Software Engineering and Advanced Applications
SEDE	19th International Conference on Software Engineering and Data Engineering
SEKE	International Conference on Software Engineering and Knowledge Engineering

Tabela A.2: List of journals in which the searches were performed.

Journal title
ACM Transactions on Software Engineering and Methodology
Automated Software Engineering
Elsevier Information and Software Technology
Elsevier Journal of Systems and Software
Empirical Software Engineering
IEEE Software
IEEE Computer
IEEE Transactions on Software Engineering
International Journal of Software Engineering and Knowledge Engineering
Journal of Software: Evolution and Process
Software Quality Journal
Journal of Software
Software Practice and Experience Journal

Tabela A.3: Search string per Search Engine.

Search Engine	Search String
Google Scholar	bug report OR track OR triage “change request” issue track OR request OR software OR “modification request” OR “defect track” OR “software issue” repositories maintenance evolution
ACM Portal	Abstract: "bug report"or Abstract:"change request"or Abstract:"bug track"or Abstract:"issue track"or Abstract:"defect track"or Abstract:"bug triage"or Abstract: "software issue"or Abstract: "issue request"or Abstract: "modification request") and (Abstract:software or Abstract:maintenance or Abstract:repositories or Abstract:repository
IEEExplorer (1)	((((((((((("Abstract": "bug report") OR "Abstract":"change request") OR "Abstract":"bug track") OR "Abstract":"software issue") OR "Abstract":"issue request") OR "Abstract":"modification request") OR "Abstract":"issue track") OR "Abstract":"defect track") OR "Abstract":"bug triage") AND "Abstract":software)
IEEExplorer (2)	((((((((((("Abstract": "bug report") OR "Abstract":"change request") OR "Abstract":"bug track") OR "Abstract":"software issue") OR "Abstract":"issue request") OR "Abstract":"modification request") OR "Abstract":"issue track") OR "Abstract":"defect track") OR "Abstract":"bug triage") AND "Abstract":maintenance)
IEEExplorer (3)	((((((((((("Abstract": "bug report") OR "Abstract":"change request") OR "Abstract":"bug track") OR "Abstract":"software issue") OR "Abstract":"issue request") OR "Abstract":"modification request") OR "Abstract":"issue track") OR "Abstract":"defect track") OR "Abstract":"bug triage") AND "Abstract":repositories)
IEEExplorer	((((((((((("Abstract": "bug report") OR "Abstract":"change request") OR "Abstract":"bug track") OR "Abstract":"software issue") OR "Abstract":"issue request") OR "Abstract":"modification request") OR "Abstract":"issue track") OR "Abstract":"defect track") OR "Abstract":"bug triage") AND "Abstract": repository)
Citeseer Library	(abstract: "bug report"OR abstract:"change request"OR abstract:"bug track"OR abstract:"issue track"OR abstract:"defect track"OR abstract:"bug triage"OR abstract: "software issue"OR abstract: "issue request"OR abstract: "modification request") AND (abstract:software OR abstract:maintenance OR abstract:repositories OR abstract:repository)
Elsevier	("bug report"OR "change request"OR "bug track"OR "issue track"OR "defect track"OR "bug triage"OR "software issue"OR "issue request"OR "modification request") AND (software OR maintenance OR repositories OR repository)
Scirus	("bug report"OR "change request"OR "bug track"OR "issue track"OR "defect track"OR "bug triage"OR "software issue"OR "issue request"OR "modification request") AND (software maintenance OR repositories OR repository) ANDNOT (medical OR aerospace)
ScienceDirect	("bug report"OR "change request"OR "bug track"OR "issue track"OR "defect track"OR "bug triage"OR "issue request"OR "modification request") AND LIMIT-TO(topics, "soft ware")
Scopus	("bug report"OR "change request"OR "bug track"OR "issue track"OR "defect track"OR "bug triage"OR "software issue"OR "issue request"OR "modification request") AND (software maintenance OR repositories OR repository)
Wiley	("bug report"OR "change request"OR "bug track"OR "issue track"OR "defect track"OR "bug triage"OR "software issue"OR "issue request"OR "modification request") AND (software maintenance OR repositories OR repository)
ISI Web of Knowledge	("bug report"OR "change request"OR "bug track"OR "issue track"OR "defect track"OR "bug triage"OR "software issue"OR "issue request"OR "modification request") AND (software maintenance OR repositories OR repository) ANDNOT (medical OR aerospace)
SpringerLink	("bug report"OR "change request"OR "bug track"OR "issue track"OR "defect track"OR "bug triage"OR "software issue"OR "issue request"OR "modification request") AND (software maintenance OR repositories OR repository) ANDNOT (medical OR aerospace)