Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"
Curso Superior de Tecnologia em Segurança da Informação

Gabriel de Godoy Vido

Kernel Hardening: Uma breve introdução a ferramentas e arquiteturas usadas em sistemas de alta criticidade

Americana, SP
2023

FACULDADE DE TECNOLOGIA DE AMERICANA "MINISTRO RALPH BIASI"

Curso Superior de Tecnologia em Segurança da Informação

Gabriel de Godoy Vido

Kernel Hardening: Uma breve introdução a ferramentas e arquiteturas usadas em sistemas de alta criticidade

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof. Marcus Vinicius Lahr Giraldi.

Área de concentração: Sistemas Operacionais

Americana, SP.

2023

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana Ministro Ralph Biasi-CEETEPS Dados Internacionais de Catalogação-na-fonte

VIDO, Gabriel de Godoy

Kernel hardening: uma breve introdução a ferramentas e arquiteturas usadas em sistemas de alta criticidade. / Gabriel de Godoy Vido – Americana, 2023.

38f.

Monografia (Curso Superior de Tecnologia em Segurança da Informação) - - Faculdade de Tecnologia de Americana Ministro Ralph Biasi – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Esp. Marcus Vinícius Lahr Giraldi

1. Segurança em sistemas de informação 2. Sistemas operacionais. I. VIDO, Gabriel de Godoy II. GIRALDI, Marcus Vinícius Lahr III. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana Ministro Ralph Biasi

CDU: 681.518.5
681.3.066

Elaborada pelo autor por meio de sistema automático gerador de ficha catalográfica da Fatec de Americana Ministro Ralph Biasi.

Gabriel de Godoy Vido

Gabriel de Godoy Vido

Kernel Hardening: Uma breve introdução a ferramentas e arquiteturas usadas em sistemas de alta criticidade

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Curso Superior de Tecnologia em Segurança da Informação pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana – Ralph Biasi.
Área de concentração: Sistemas Operacionais

Americana, 17 de junho de 2023

Banca Examinadora:

Marcus Vinicius Lahr Giraldi (Presidente)
Especialista
FATEC – Faculdade de Tecnologia de Americana

Henri Alves de Godoy (Membro)
Doutor
FATEC – Faculdade de Tecnologia de Americana

Elton Rafael Mauricio da Silva Pereira (Membro)
Mestre
FATEC – Faculdade de Tecnologia de Americana

Objetivo

O Trabalho busca mostrar as principais técnicas que auxiliam a manter um servidor seguro como também auxiliar na parte de configuração dos serviços que irão ser executados nesse servidor. Propõe de maneira simplificada mostrar incidentes de segurança que se basearam em vulnerabilidades causadas por motivos da criação desse trabalho, como, má configuração de serviços, sistemas operacionais desatualizados e arquivos de backups que foram esquecidos.

Justificativa

Estudos recentes mostram que a frequência de ataques está aumentando devido às novas tecnologias que estão surgindo, os atacantes estão inovando em seus meios de ataques e novos métodos e técnicas devem surgir para atender a essa necessidade do mercado.

Existem muitos trabalhos referentes ao tema de *Kernel Hardening* no exterior. Contudo no Brasil essa linha de pesquisa não está sendo muito explorada. Este presente trabalho tem o objetivo de aumentar o arsenal de pesquisa a respeito de *Kernel Hardening* no cenário cibernético brasileiro.

Hipótese

O autor acredita que o desenvolvimento de novos trabalhos a respeito do tema é uma parte essencial para conscientização, incentivo e aplicação da temática e auxiliar a pesquisa na implementação do tema no cotidiano dos administradores de sistema.

Problema

Nos últimos anos os ataques cibernéticos têm aumentado devido a diversos motivos, entre eles o maior causador são ataques que utilizam algum serviço, framework ou sistema operacional desatualizado e / ou mal configurado.

Baseado nisso atualmente os administradores de sistemas, estão começando a criar uma mentalidade mais madura, a respeito de quais softwares utilizar no sistema, como manter seu sistema operacional atualizado, com os *patches* mais recentes de segurança e também estão aprimorando as boas práticas de configuração em seus serviços com o objetivo de diminuir os vetores de onde atacantes podem utilizar para seus ataques.

Metodologia

Este trabalho é focado principalmente na metodologia de pesquisa de revisão bibliográfica, como livros, artigos e textos já publicados e verificados.

Utilizada também a pesquisa experimental, para a realização de testes focados na confirmação das ideias desse trabalho.

Resumo

Neste trabalho serão abordadas algumas ferramentas utilizadas por administradores de sistemas que auxiliam constantemente a monitorar o funcionamento normal esperado dos serviços em execução em seu sistema operacional. Será abordado também algumas praticas que administradores de sistemas utilizam para verificar a integridade de seus serviços, visando descobrir se algum está comprometido ou não, nisso será demonstrado quais módulos do sistema operacional Linux implementam alguns modelos de permissionamento como MAC através do SELinux e AppArmor. Além disso será tratado de práticas para proteção de binários e executáveis no sistema operacional para evitar possíveis vetores que um atacante poderia utilizar, como injeção de variável PATH para Escalamento de Privilégio.

Palavras-chave: Modelos de Permissionamento, SELinux, AppArmor

Abstract

This work will address some tools used by system administrators that constantly assist in monitoring the expected normal functioning of services running on their operating system. It will also cover some practices that system administrators employ to verify the integrity of their services, aiming to identify any compromises. Additionally, it will demonstrate which modules of the Linux operating system implement permission models such as MAC through SELinux and AppArmor. Furthermore, it will discuss practices for protecting binaries and executables in the operating system to prevent potential vectors that an attacker could exploit, such as PATH variable injection for Privilege Escalation.

Keywords: Access-Controls, SELinux, AppArmor

Lista de Figuras

1. Introdução

Conforme as ameaças que atacantes impõem nos tempos modernos, é inevitável que um ou outro servidores acabem sendo comprometidos por tais atacantes, sendo assim é importante ter um sistema *hardenizado* para a minimizar o impacto de tal atacante o máximo possível.

Baseado nisso é de extrema importância que um administrador de sistema tenha ferramentas que o auxilie na boa gestão dos serviços em execução no sistema operacional para garantir sua integridade, sendo através de monitoramento dos serviços ou atualizando seus serviços e programas para evitar que os invasores utilizem de vulnerabilidades e *exploits* de versões antigas.

Um administrador deve ter noção do ambiente em que está sendo executado seu sistema operacional, caso seja um ambiente que necessite extrema segurança e confidencialidade como ambientes médicos, governamentais, e de alta tecnologia, é recomendado o uso de modelos de permissionamento mais “Rígidos” como o MAC ou RBAC, contudo caso seja um ambiente que tolerável e que permita um nível de acesso maior, onde os usuários possam controlar os níveis de permissão, é recomendado o uso do DAC.

2. Atualização do Sistema Operacional

A tarefa principal de um administrador de sistema é manter um ambiente de trabalho e um sistema operacional seguro e mantê-lo atualizado através de *patches* com o objetivo de corrigir vulnerabilidades descobertas em versões antigas do sistema operacional. Porém realizar essas tarefas acaba se tornando algo repetitivo e cansativo, assim é interessante a ideia de implementar as atualizações de forma automática ou com certa periodicidade para evitar a possibilidade de erro humano, existem ferramentas que auxiliam na realização desta tarefa como a ferramenta ***cron***[1].

Figura 1: Demonstra um exemplo de como seria feito um agendamento de atualização no sistema operacional através da ferramenta ***cron***

```
0 22 * * sat root /usr/bin/apt update -q -y >> /var/log/apt/scheduledupdates.log
20 22 * * sat root /usr/bin/apt upgrade -q -y >> /var/log/apt/scheduledupdates.log
```

[1] https://manpages.debian.org/unstable/cron/cron.8.en.html

Fonte: Autor

A Figura 1 demonstra a criação de uma rotina no */etc/crontab* Tomando a primeira linha como exemplo, o primeiro caractere (0) é o campo que informa os minutos, o segundo caractere (22) é respectivo a horas, o terceiro caractere (*) é referente ao mês, o quarto caractere (*) é respectivo aos meses, o quinto caractere (sat) diz respeito há quais dias da semana devem ser feitas essas tarefas. Após as definições de horário vem o nome do usuário que dará as permissões para o executável realizar a tarefa (root), em seguida vem o comando que será executado (/usr/bin/apt update -q -y), após a execução do comando determinado, o resultado é exibido. Neste caso , o resultado do comando está sendo redirecionado (representado pelos sinais >>) para um arquivo de registro que poderá ser acessado pelo administrador (/var/log/apt/scheduleupdates.log).

Ou seja, o comando será executado às 22 horas (22) aos 0 minuto (*), nos doze meses (*), durante todos os dias do mês (*), devido a *flag* (sat) o comando será executado apenas em sábados, onde será executado um update no modo silencioso (-q), serão aceitas todas as requisições feitas (-y) e por fim o output desse comando será adicionado em um arquivo de log.

É importante saber como referenciar o comando e o seu executável, pois caso isso não seja definido corretamente ou aplicado corretamente, é possível que um atacante consiga manipular o caminho que o ***cron*** procura, através de executáveis e realizar uma tarefa que não é a planejada, podendo assim causar dano no sistema operacional, pois essa tarefa seria executada como *root*.

Contudo, novas atualizações vão chegando, assim novas vulnerabilidades vão surgindo, tornando o processo de atualização contra intuitivo e grande maioria das vulnerabilidades que estão na Internet são antigas e só funcionam em sistemas operacionais legados, fazendo assim um grupo minoritário de vulnerabilidades que funcionem em sistemas operacionais modernos e atualizados.

> *“Às vezes, os patches de software introduzem novos problemas de segurança e suas próprias fraquezas.*
> *No entanto, a maioria das explorações tem como alvo vulnerabilidades mais antigas que são amplamente conhecidas. Estatisticamente falando, você está muito melhor com sistemas que são atualizados regularmente.*
> *Certifique-se de que seja feito de forma metódica e consistente”* (NEMETH; SNYDER; HEIN; WHALEY; MACKIN, 2018, P.988)

3. Monitoramento de processos e verificação de integridade

Monitoramento é uma etapa essencial para manter um sistema seguro e disponível através do monitoramento de serviços, monitorar serviços garante que eles estejam funcionando e garantindo a disponibilidade, confidencialidade e integridade, a etapa de monitoramento auxilia na parte de detecção de invasões realizadas por atacantes onde a própria presença do atacante, muda o funcionamento casual dos serviços.

> *"O monitoramento é parte integrante de qualquer política de segurança por vários motivos. Entre elas, que o objetivo da segurança geralmente não se restringe a garantir a confidencialidade dos dados, mas inclui também garantir a disponibilidade dos serviços"* (HERTZOG; MAS, 2018, P.410)

Existem várias ferramentas para realizar as atividades de monitoramento em sistemas operacionais, referentes ao uso de recursos como CPU, RAM, uso de disco, uso de rede etc.

Existem ferramentas que realizam as funções de registro de *log* no sistema, seguem alguns exemplos abaixo.

A ferramenta ***top***[2] realiza *logs* em tempo real, sobre todos os processos que estão rodando no sistema e exibe |informações a respeito do uso de CPU, RAM, uso de disco, quais processos estão sendo executados, quais estão parados, quais estão dormindo; ou seja, aguardando ser ativado e qual o comando executado por aquele processo.

A ferramenta que registra *logs* no sistema e salva de arquivos é o ***syslog***[3], essa ferramenta funciona em qualquer distribuição Linux. Ela, diferentemente da ferramenta ***top***, realiza uma coleta passiva de dados, assim os dispositivos que estão configurados com o cliente de ***syslog***, enviam passivamente os dados para um servidor central na rede, que realiza a função de *logging*. A ferramenta funciona em

---

[2] https://manpages.ubuntu.com/manpages/xenial/man1/top.1.html

[3] https://manpages.debian.org/jessie/manpages-dev/syslog.3.en.html

um estilo de cliente-servidor, sendo assim, os dados coletados dos clientes são mandados para um servidor central, processados e armazenados.

A ferramenta que realiza a função de verificação de *logs* é a ferramenta ***logcheck***[4], onde ela realiza a função de verificação de integridade dos arquivos de *log*, realizados no sistema Linux de hora em hora, a ferramenta possui 3 modos de configuração, onde cada um muda a intensidade de detalhamento. A ferramenta também filtra as mensagens de *log*, por exemplo: uma tentativa de quebra de senhas seria armazenada em um diretório e uma mensagem de segurança a respeito de *firewalls* seria armazenada em outro diretório.

A ferramenta voltada para a parte de proteção a invasões é a ferramenta ***fail2ban***[5], a ferramenta trabalha principalmente em evitar ataques de força bruta, colocando limites no número de tentativas de Login dos usuários. a ferramenta funciona em um modelo de cliente-servidor, onde o cliente possui uma aplicação instalada e envia os dados das requisições negadas para o servidor, que por sua vez, as armazena para futuro uso. Caso sejam detectadas 5 tentativas falhas no cliente que possui a aplicação, o IP que realizou as tentativas será adicionado a uma *blacklist* e assim proibido de tentar de se conectar novamente.

Uma outra ferramenta, contudo comumente utilizada por invasores, que consegue realizar um monitoramento superficial dos processos porém sem o permissionamento necessário é a ferramenta ***pspy***[6], ela consegue realizar uma verificação de todos os processos que estão em execução no sistema operacional através do acompanhamento dos arquivos no diretório ***/proc*** do Linux, *bypassando (driblando)* a necessidade de permissão para a visualização desses processos.

Atualmente muitos ataques na Internet se devem através de arquivos modificados e assim conseguindo acessos não autorizados através de código malicioso ofuscado nesses tais arquivos.

Segundo CISO Advisor (Redação; 2022):

"*As amostras observadas com mais frequência incluem vírus, worms, cavalos de Troia (trojan), ransomware, spyware e rootkits.*".

---

[4] https://linux.die.net/man/8/logcheck

[5] https://www.fail2ban.org/wiki/index.php/Main_Page

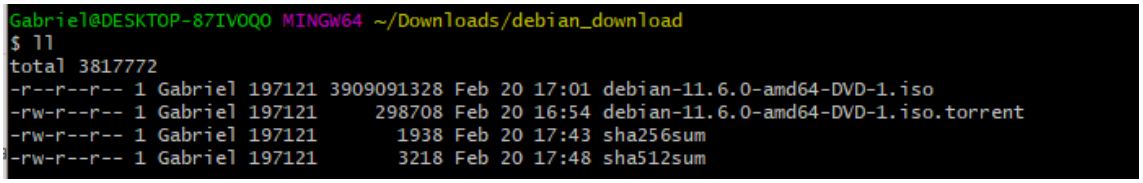
[6] https://github.com/DominicBreuker/pspy

Tomando como base o *malware* Cavalo de Tróia que se instala em arquivos legítimos e é executado durante a execução do arquivo original, uma das maneiras de prevenir esse tipo de ameaça é através de verificação de integridade dos arquivos, por exemplo: O site do sistema operacional *Debian* oferece o serviço de download de suas imagens e *hashes* necessários para verificar a integridade do arquivo que foi baixado.

Isso serve como pilar para garantir 3 coisas a respeito sobre o arquivo baixado:

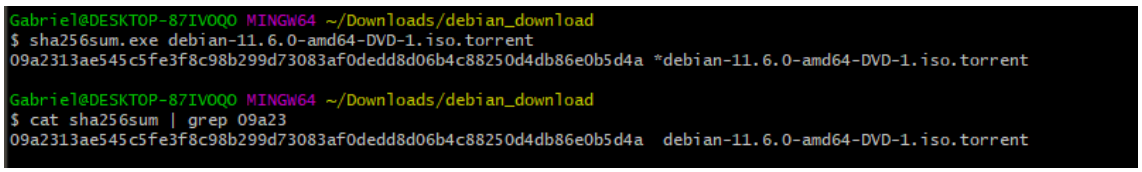
1. **Garantia que o arquivo não foi corrompido**: caso o arquivo esteja corrompido o *hash* gerado pela função será diferente da provida pelo servidor, outro exemplo é a utilização dessas funções criptográficas para gerar *checksums* de arquivos de backup, onde o primeiro será tomado como base e caso esteja diferente há uma grande chance do backup estar corrompido
2. **Detectar se houve alterações não autorizadas**: extremamente útil ao passar arquivos sensíveis e que não devem ser alterados sem permissão prévia
3. **Redução e risco de *malware*:** Caso o arquivo que tenha sido baixado não tenha o mesmo *hash* que o servidor proveu existe há possibilidade que esse arquivo esteja corrompido ou tenha sido modificado para causar algum mal ao computador, comportamento comum de *malwares* como *trojan*, *bloatware*, *spyware* etc…

Figura 2: Visualização de um arquivo *.torrent*, o arquivo ISO (imagem do sistema operacional) e 2 arquivos com *hashes* de verificação de integridade (sha256sum e sha512sum)



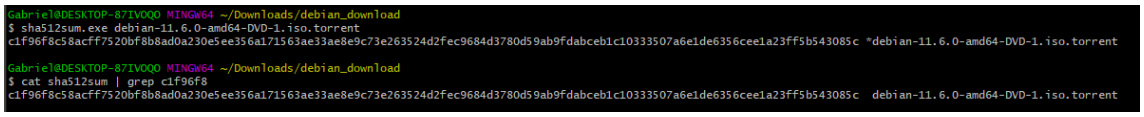
Fonte: Autor

Figura 3: Verificação de integridade dos arquivos .torrent com os hashes de SHA256



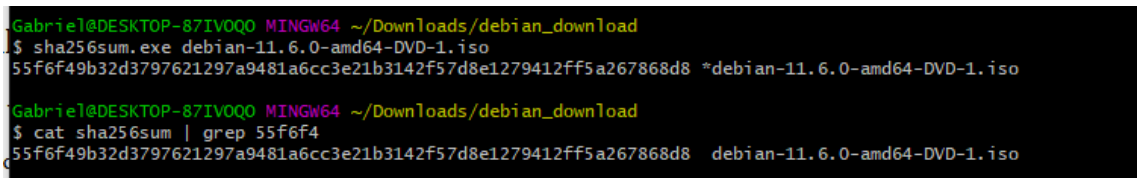
Fonte: Autor

Figura 4: Verificação de integridade dos arquivos .torrent com os hashes de SHA512



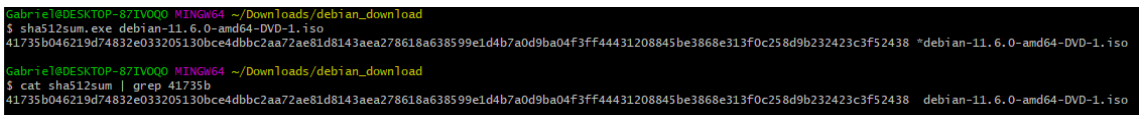
Fonte: Autor

Figura 5: Verificação de integridade dos arquivos ISO com os hashes de SHA256



Fonte: Autor

Figura 6: Verificação de integridade dos arquivos ISO com os hashes de SHA512



Fonte: Autor

A figura 3 demonstra a verificação dos arquivos .torrent com o algoritmo SHA256. A figura 4 demonstra a verificação dos arquivos .torrent com o algoritmo SHA512. A figura 5 demonstra a verificação dos arquivos .iso com o algoritmo SHA256. A figura 6 demonstra a verificação dos arquivos .iso com o algoritmo SHA512.

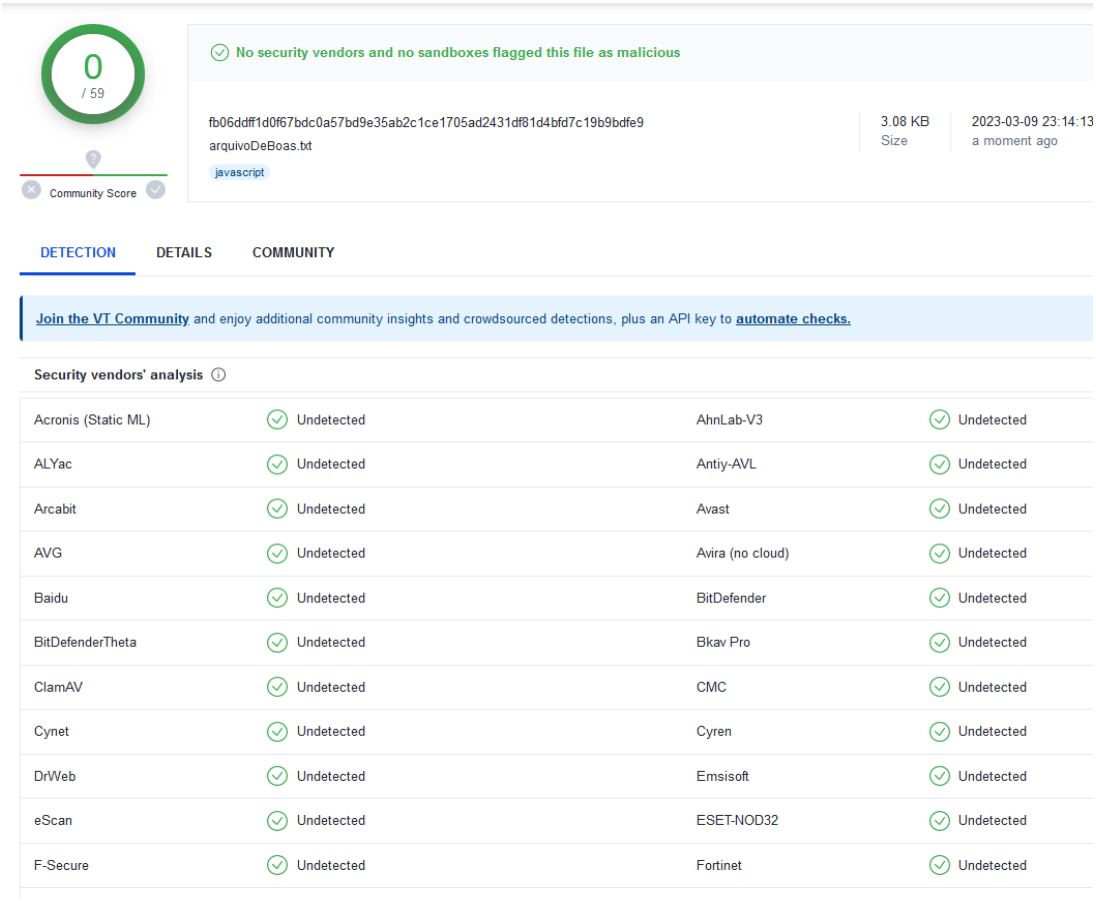
Existem recursos que auxiliam na detecção e análise de arquivos e também utilizam *hashes* de arquivo para detectar assinaturas similares e consequentemente realizar sua classificação como malicioso ou inofensivo.

Entre esses sites está o VirusTotal[7], onde ele disponibiliza serviços com o objetivo de auxiliar a comunidade de segurança da informação, onde esses serviços se especializam em detecção de código malicioso em arquivo que forem submetidos, ele realiza isso submetendo o arquivo que foi provido pelo cliente em vários softwares anti-vírus, após terminar em todos os *scanners* ele captura o resultado e demonstra os valores obtidos.

A figura 7 representa um *scan* realizado em 09/03/2023 às 20:34 UTC-8 sobre um arquivo de texto normal e a figura 8 representa um *scan* em um *trojan* de Windows XP às 20:47. Algo interessante ao notar que as datas ao lado representam a assinatura (*signature*) do arquivo, onde o primeiro arquivo não possuía nenhuma *signature* presente nos bancos de dados dos softwares antivírus nem do VirusTotal, contudo o segundo arquivo já possui uma *signature* presente onde a primeira foi detectada em 11/02/2022.

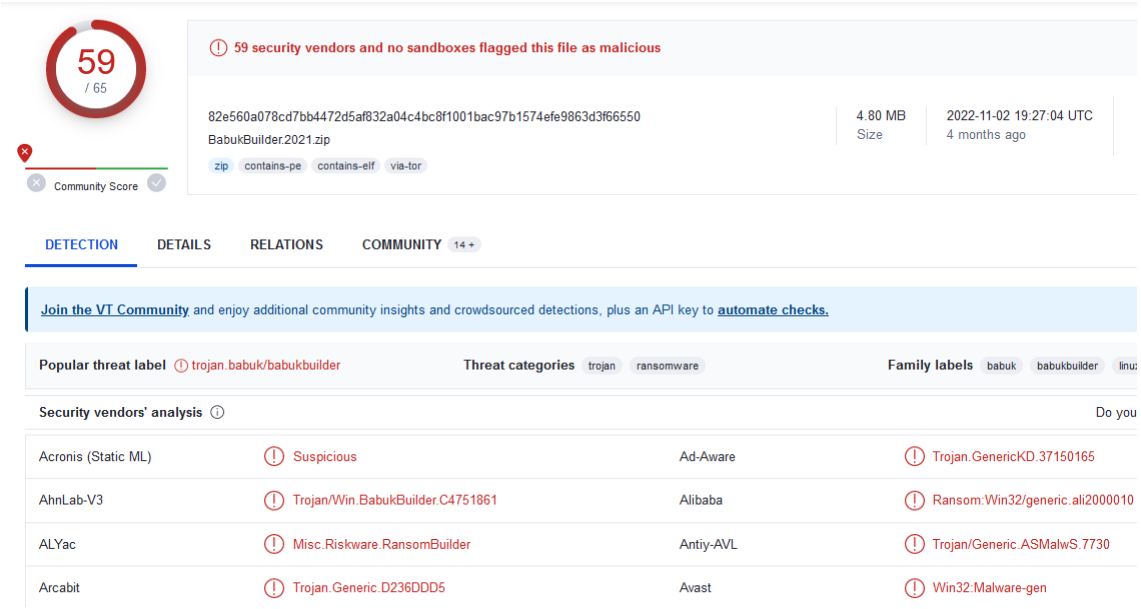
[7] https://www.virustotal.com/gui/home/upload

Figura 7: Upload de um arquivo aparentemente inofensivo e os resultados dos softwares anti-virus



Fonte: Autor

Figura 8: Upload de um arquivo possivelmente infectado e os resultados dos softwares anti-virus



Fonte: Autor

Contudo existem críticas a respeito dessa plataforma, como coloca Wang, Song, Yang e Peng:

> "*Infelizmente, o VirusTotal funciona como uma caixa preta, e não é bem compreendido como a VirusTotal e os seus vendedores geram as avaliações/tags para um determinado URL ou arquivo. Isto leva a questões críticas: são estas avaliações mesmo fiéis? Os investigadores estão a usar o VirusTotal da forma correta?" (Opening the Blackbox of VirusTotal: Analyzing Online Phishing Scan Engines; WANG; SONG; YANG; PENG; 2019; P.478)*

4. Controle de acesso

O controle de acesso é essencial em servidores Linux porque ajuda a garantir a segurança e a integridade do sistema e de seus dados. Ele permite que o administrador controle quem tem acesso a recursos específicos e qual nível de acesso eles possuem. Sem ele, os usuários não autorizados podem ter acesso a informações confidenciais ou realizar atividades maliciosas, como excluir arquivos importantes, instalar *malware* ou roubar dados, como é dito pela ISC³, 2023.

> "Os Controles de Acesso ajudam os gerentes a limitar e monitorar o uso dos sistemas em nível de usuário, e geralmente são predefinidos com base no nível de autoridade ou associação

> a grupos. Você compreenderá os diferentes sistemas de controle de acesso e como eles devem ser implementados para proteger o sistema e os dados, utilizando os diferentes níveis de confidencialidade, integridade e disponibilidade.”

Os servidores Linux são frequentemente usados para hospedar aplicativos e dados críticos, incluindo informações financeiras, dados de clientes e propriedade intelectual. Qualquer violação ou acesso não autorizado a essas informações pode levar a consequências graves, incluindo perdas financeiras, danos à reputação e consequências legais. O controle de acesso ajuda a prevenir tais violações, aplicando políticas estritas de acesso que limitam o acesso apenas a usuários e processos autorizados.

Ele também permite uma gestão mais fácil de permissões de usuários e níveis de acesso. O administrador pode atribuir direitos de acesso específicos a cada usuário ou grupo e atualizá-los conforme necessário. Isso pode ajudar a garantir que os usuários tenham acesso apenas aos recursos necessários para desempenhar suas funções e não mais do que isso. Também ajuda a reduzir o risco de perda ou corrupção de dados acidental ou intencional.

Em geral, o controle de acesso é um aspecto crucial da segurança de servidores Linux. Ele ajuda a manter a confidencialidade, integridade e disponibilidade do sistema e de seus dados, protegendo contra acesso não autorizado e violações de dados. Ao implementar políticas e ferramentas adequadas de controle de acesso, os administradores podem garantir que seus servidores Linux sejam seguros e protegidos contra ameaças potenciais e atualmente existem múltiplos modelos de permissionamento e controle de acesso, sendo os mais comumente encontrados: DAC, MAC e RBAC.

Entre os diversos tipos de modelos permissivos, o DAC é um tipo de mecanismo de controle de acesso que permite aos usuários ter controle total sobre seus arquivos e recursos. O proprietário ou criador dos arquivos decide quem pode acessá-los e quais as que ações podem ser realizadas com eles. Ele é utilizado em vários sistemas operacionais como o Windows e sistemas baseados em Unix como Linux, onde o mesmo é o modelo de permissão ativado por *default*.

O modelo de permissionamento do DAC é constituído em ACLs (*Access Control Lists/Listas de Controle de Acesso*), onde os usuários presentes nas ACLs terão ou não os direitos respectivos aquele arquivo ou recurso, dentre eles existem os direitos de ler, escrever, executar e excluir. Os usuários têm controle total sobre o arquivo e podem delegar outras permissões.

Uma das principais vantagens desse modelo é sua simplicidade na implementação e uso, talvez sendo a mais importante. Os usuários podem entender facilmente a quem pertence o arquivo ou recurso e suas permissões. Porém ao aplicar esse modelo simplista de permissões em alta escala pode acabar sendo um problema, pois existem vários cenários onde um arquivo ou recurso necessita de permissões especiais, contudo o DAC não consegue atender a essa necessidade.

Outro problema com o modelo DAC é o impacto resultante de ataques, onde no caso um atacante possua acesso a conta um usuário, ele poderá controlar todos os arquivos e recursos que aquele usuário tem permissão, pior ainda caso essa conta possua privilégios de administrador, nesse caso o atacante poderá efetivamente controlar o sistema operacional inteiro, afetando sua disponibilidade, integridade e confidencialidade. Uma maneira que os administradores de sistema encontraram de solucionar esse problema foi através da implementação de outros sistemas de permissionamento como por exemplo o MAC (*Mandatory Access Control*).

Outro modelo de permissionamento existente é o modelo de MAC, onde ele por si só é completamente diferente do DAC, onde as permissões em si são providas pelo sistema sem a necessidade direta de um administrador eliminando a possibilidade de erro humano. Ele delega as permissões de acesso, modificação e execução através de listas contendo informações de segurança específicas para cada usuário e recursos do sistema. Todos os usuários do sistema devem seguir as regras criadas pelo MAC independente da hierarquia de permissão, podendo ser tanto um usuário quanto administrador.

A implementação mais comum que pode ser vista do MAC é através de um sistema de multi-camadas (MLS), onde o usuário solicitante da ação deve passar por múltiplos níveis de permissionamento para conseguir realizar sua ação. Normalmente as regras que serão elaboradas e implementadas no servidor central que irá

armazenar as regras do modelo MAC são regras baseadas no modelo de negócio da organização que está instalando esse permissionamento, a configuração das listas que serão implementadas e contém as informações de segurança é feita pelo administrador de sistema onde ele irá atribuir o respectivo permissionamento para o recurso, por exemplo: em um sistema onde há 3 níveis de permissionamento: privado, secreto e ultrasecreto, onde cada usuário com o seu rótulo respectivo conseguira acessar apenas os recursos com o mesmo nível de permissionamento, ou seja, um usuário privado somente conseguirá utilizar recursos de rótulo privado e um usuário ultra-secreto somente poderá utilizar recursos de nível ultra-secreto.

> "*O MAC recomenda que além das permissões associadas à sua identidade, cada objecto dentro do ambiente tem a sua própria tag associada ao objeto. Agora, com base nas permissões que me foram concedidas, talvez associadas ao meu papel ou grupo, só posso realizar ações sobre objetos com as tags se uma política explícita me conceder acesso ao objecto. Inversamente, uma política pode ser escrita com base em tags para me negar explicitamente (ou o meu papel/grupo)" (MAC vs DAC vs RBAC; Davis; 2014)*

Contudo devido à complexidade da aplicação efetiva desse modelo de permissionamento, ele normalmente é aplicado em sistemas que possuem informações de extrema sensibilidade como sistemas governamentais, hospitais, tecnologia e serviços de propriedade intelectual.

Um outro modelo também muito utilizado, inclusive utilizado pelo ja mencionado SELinux, é o RBAC , cuja sua caracteristica é definida por delegar acesso baseado em cargos, onde tais usuários pertencentes a certos cargos herdam as permissões respectivas.

Interpretado de Danny Baier: Inicialmente o RBAC foi desenvolvido por uma equipe de desenvolvedores voltados para segurança, entre eles "Ravi S. Sandhu' , Edward J. Coynek , Hal L. Feinsteink and Charles E. Youman". Eles publicaram um papel chamado "Role-Based Access Control Models" onde descrevia os conceitos básicos para estabelecer um modelo de controle baseado em cargos, desde a

descrição dos usuários, cargos e suas permissões. A ideia do RBAC é implementar o MAC através de uma abordagem diferente

Atualmente o RBAC é utilizado em algumas ferramentas/serviços, entre eles:

- SELinux
- Kubernetes
- Algumas distribuições Unix: Red Hat, CentOS
- Windows Active Directory
- AWS: *Amazon Web Services*

Concluindo, tanto o controle de acesso obrigatório (MAC) quanto o controle de acesso discricionário (DAC) são modelos de segurança importantes usados para proteger informações e recursos confidenciais.

O MAC é um modelo de controle de acesso estrito em que as decisões de acesso são baseadas em regras e políticas predefinidas definidas pelo administrador do sistema. Isso significa que os usuários têm controle limitado sobre seus direitos e permissões de acesso e só podem acessar recursos explicitamente autorizados pelo sistema. O MAC é comumente usado em ambientes de alta segurança, como organizações governamentais e militares.

Por outro lado, o DAC é um modelo de controle de acesso mais flexível que permite aos usuários determinar quem pode acessar seus recursos. No DAC, os usuários têm mais controle sobre seus direitos e permissões de acesso e podem conceder ou negar acesso a recursos específicos. Esse modelo é comumente usado em ambientes menos sensíveis, como empresas e dispositivos pessoais.

Em última análise, a escolha entre MAC e DAC depende dos requisitos de segurança específicos do sistema ou organização. Embora o MAC forneça um nível mais alto de segurança, ele pode ser menos flexível e mais difícil de gerenciar. O DAC, por outro lado, oferece mais flexibilidade e facilidade de uso, mas pode não ser adequado para ambientes de alta segurança. Em resumo, tanto o MAC quanto o DAC têm suas próprias vantagens e desvantagens, e a decisão sobre qual modelo usar

deve ser baseada nas necessidades específicas de segurança da organização ou sistema.

Vale a pena notar que a implementação de um não elimina obrigatoriamente a existência do outro no sistema operacional, na prática o que atualmente é mais implementado é uma combinação dos dois, onde os processos/serviços que são de alta criticidade são confinados e processados pelas ferramentas que estão implementando o MAC, e aqueles que não são de alta criticidade e podem executar sem a necessidade de um supervisionamento extensivo são processados apenas pelo DAC.

5. Binários, Executáveis e Criptografia de Disco

Arquivos binários ou executáveis são programas que são executados no sistema operacional, por padrão, existem duas formas de se executar um programa:

- Seu caminho absoluto: caso um usuário queira executar um binário como */usr/bin/ping* ele poderia executar o binário com */usr/bin/ping,* executando um programa da forma considerada segura.

- Referencia no *$PATH:* aqui um administrador pode referenciar os arquivos no sistema operacional para apenas invoca-los com “apelidos”, por exemplo: caso o usuário pedro tenha um arquivo binário no seu *home directory* chamado “teste”, o administrador poderia colocar o caminho absoluto do arquivo na variavel *PATH,* onde caso o usuario precise executar o arquivo ele apenas digitaria “teste” no terminal ao invés de “*./home/pedro/teste*”, como é dito por Vickie Li, 2020

> “*Por exemplo, o comando ls é usado para listar os arquivos no diretório atual. E o binário para o comando ls geralmente está localizado no diretório /bin. Mas você não precisa digitar /bin/ls para executar o comando, porque o diretório /bin é adicionado à sua variável PATH. Quando você digita um nome de comando sem especificar um caminho absoluto, seu shell irá procurar em cada diretório na lista*

> *PATH por um arquivo executável com esse nome. O shell então executará o primeiro programa correspondente que encontrar.*"

Contudo existem alguns perigos por exemplo: um administrador logado como root quer criar uma tarefa para que apenas o executável ***tar*** seja executado por qualquer usuário e com suas permissões, contudo ele não especifica o caminho absoluto, invés disso apenas coloca "tar" para os usuários em ***/etc/sudoers*** executarem , a tarefa agendada será executada, contudo ela esta vulnerável a um ataque de *privilege escalation*, onde um atacante com acesso a uma conta de usuário poderia criar um arquivo malicioso chamado "tar" e caso o atacante colocasse esse arquivo malicioso em */tmp* e acrescentasse na variável $PATH o caminho de */tmp* esse arquivo com código malicioso seria executado com as permissões de administrador. Devido a isso é extremamente importante sempre utilizar caminhos absolutos e verificar se os processos em execução ou arquivos que serão executados não estão vulneráveis a esse tipo de ataque.

Atualmente, segundo a Kaspersky (Redação; 2018) "O roubo de informações ou roubo digital pode ocorrer de várias maneiras. Onde as mais comuns são:

1. **Engenharia Social:** A forma mais comum de engenharia social é o *phishing*. Onde ele ocorre quando um invasor se disfarça como uma entidade confiável para enganar a vítima a abrir um e-mail, mensagem de texto ou mensagem instantânea. Os usuários que caem em ataques de *phishing* são uma causa comum de roubo de dados.
2. **Senhas Fracas:** Usar uma senha fácil de adivinhar, ou usar a mesma senha para várias contas, pode permitir que invasores obtenham acesso aos dados. Hábitos de senha ruins - como escrever senhas em um pedaço de papel ou compartilhá-las com outras pessoas - também podem levar ao roubo de dados.
3. **Vulnerabilidades de Sistema:** Aplicativos de software mal escritos ou sistemas de rede mal projetados ou implementados criam vulnerabilidades que *hackers* podem explorar e usar para roubar dados. O *software* antivírus desatualizado também pode criar vulnerabilidades.
4. **Ameaças Internas:** Os funcionários que trabalham para uma organização têm acesso às informações pessoais dos clientes. Funcionários mal-intencionados ou contratados descontentes podem copiar, alterar ou roubar dados. No entanto, ameaças internas não se limitam necessariamente a funcionários atuais. Eles também podem ser ex-funcionários, contratados ou parceiros que têm acesso

aos sistemas ou informações sensíveis de uma organização. As ameaças internas estão aumentando.

5. **Erro Humano:** Violações de dados não precisam ser o resultado de ações maliciosas. Às vezes, eles podem ser o resultado de um erro humano. Erros comuns incluem o envio de informações sensíveis para a pessoa errada, como enviar um *e-mail* por engano para o endereço incorreto, anexar o documento errado ou entregar um arquivo físico para alguém que não deveria ter acesso à informação. Alternativamente, um erro humano poderia envolver uma configuração incorreta, como um funcionário deixando um banco de dados que contém informações sensíveis online sem quaisquer restrições de senha.
6. **Downloads Maliciosos:** Um indivíduo pode baixar programas ou dados de sites comprometidos infectados por vírus como *malware*. Isso dá a criminosos acesso não autorizado aos seus dispositivos, permitindo que eles roubem dados.
7. **Ações Físicas:** Algum roubo de dados não é o resultado de crime cibernético, mas de ações físicas. Isso inclui o roubo de documentos ou dispositivos como *laptops*, telefones ou dispositivos de armazenamento. Com o trabalho remoto cada vez mais difundido, a possibilidade de dispositivos se perderem ou serem roubados também aumentou. Se você está trabalhando em um local público como uma cafeteria, alguém pode observar sua tela e teclado para roubar informações, como suas informações de Login. *Skimming* de cartão - onde os criminosos inserem um dispositivo em leitores de cartões e caixas eletrônicos para colher informações de cartão de pagamento - é outra fonte de roubo de dados.
8. **Banco de dados ou problemas nos servidores:** Se uma empresa que armazena suas informações for atacada devido a um problema de banco de dados ou servidor, o atacante pode acessar as informações pessoais dos clientes.
9. **Domínios Públicos:** Muitas informações podem ser encontradas no domínio público - ou seja, por meio de pesquisas na Internet e verificação de postagens de usuários em redes sociais.

(*What is data theft and how to prevent it; Kaspersky; 2018*)"

Dito isso é possível perceber a imensa superfície de ataque que os invasores possuem para roubar dados de organizações, empresas e dos usuários, contudo os ataques vão se aprimorando os profissionais de defesa também se atualizam as ameaças iminentes para proteger esses dados, sejam através novas tecnologias como *firewalls* de nova geração, *IPSs*, *IDSs, Zero trust methodology* etc…

Porém uma prática que não tem sido muito comentada e que pode eliminar efetivamente duas ou três das superfícies de ataques citadas pela Kaspersky é a criptografia de disco.

A criptografia de disco atualmente pode ser realizada através de 2 métodos:

- Criptografia de disco através de software: A Criptografia de disco realizada por software instalado no sistema operacional, atualmente existem ferramentas para fazer isso como Veracrypt, Bestcrypt, Bitlocker e Truecrypt
- Criptografia de disco através de hardware: A Criptografia de hardware funciona como um *middleware* entre o disco instalado no sistema operacional e a BIOS, onde ao bootar o sistema operacional uma *BIOS* modificada será iniciada e ira requisitar uma senha para decriptar o disco rígido, segundo a Kingston (Maio 2021):

  > "*A criptografia de hardware fica entre o sistema operacional instalado no disco e o BIOS do sistema. Quando o disco é criptografado pela primeira vez, uma chave de criptografia é gerada e armazenada na memória flash NAND. Quando o sistema é inicializado pela primeira vez, um BIOS personalizado é carregado e solicitará uma frase secreta do usuário. Uma vez que a frase secreta é inserida, o conteúdo do disco é descriptografado e o acesso ao sistema operacional e aos dados do usuário é concedido*"

A criptografia de disco consegue proteger uma empresa contra certos ataques físicos e até mesmo ataques realizados de forma interna, que podem ser realizados por invasores, o ataque mais comum que pode ser citado é chamado de "*dumpster diving",* onde é uma técnica utilizada por atacantes para descobrir informações dos usuários ou de uma empresa através do descarte de lixo de documentos confidenciais ou discos rígidos, essas informações podem incluir senhas, nomes de usuários, endereços de e-mail, informações financeiras ou outras informações confidenciais que possam ser utilizadas para obter acesso a sistemas ou redes de computadores. Por si só a criptografia de disco consegue anular o ataque pois caso o atacante consiga um disco rígido descartado os dados presentes nele estarão encriptados, Isso significa que, mesmo que o *hacker* encontre documentos ou arquivos em um lixo ou em um dispositivo de armazenamento descartado, essas informações estarão criptografadas e, portanto, inacessíveis para o invasor.

Além disso, a criptografia de disco pode ajudar a prevenir outros tipos de ataques, como roubo ou perda de dispositivos de armazenamento, já que, mesmo que esses dispositivos caiam em mãos erradas, as informações armazenadas permanecerão protegidas.

Como dito por Gerasimo J. Stellatos (*The Impact of Full Disk Encryption on Digital Forensics; 2008*):

> "*Em um caso recente, a Alfândega dos EUA observou possível pornografia infantil no laptop de Sebastien Boucher, mas não percebeu que ela estava armazenada em um contêiner criptografado com Pretty Good Privacy (PGP) que estava aberto e atribuído à letra de unidade "Z" no momento da inspeção. O laptop foi desligado e foi criada uma cópia forense do disco rígido, mas os examinadores forenses não conseguiram abrir o contêiner criptografado com PGP. Uma intimação do Grande Júri para compelir Boucher a fornecer sua senha para descriptografar os dados foi negada com base no argumento de que violaria seu direito à Quinta Emenda contra a autoincriminação."*

O processo da criptografia funciona através da criptografia dos *bits* de dados armazenados em blocos ou um volume de disco, o processo da criptografia é realizada através de software ou hardware onde uma senha é escolhida pelo operador para criptografar os *bits* no disco, importante notar que todas os dados presentes nos volumes serão criptografados, contudo há um único setor do disco que não será criptografado, que será o setor contendo o MBR (*Master Boot Record*) ou o GPT (*GUID Partition Table*) caso implementado.

As cifras utilizadas para encriptar os *bits* variam de acordo com o sistema operacional e qual ferramenta será utilizada, por exemplo: atualmente em sistemas UNIX é utilizado a ferramenta LUKS (*Linux Unified Key Setup-on-disk-format*) que utiliza por padrão a cifra *aes-xts-plain64* e onde o tamanho padrão da chave utilizada para criptografar o disco é de 512 *bits.* Contudo isso pode variar de acordo com as cifras presentes no sistema operacional, podendo ser mudadas a qualquer instante, como é dito por Akito em Stack Exchange, 2019.

6. Modulos de Segurança

Dentre os outros módulos existentes que implementam o MAC em sistemas operacionais, existe um que consegue ser de certa forma simples e oferece a mesma flexibilidade e segurança que o MAC propõe, e ele é o AppArmor.

O AppArmor foi inicialmente desenvolvido como um módulo para o sistema operacional *Immunix* (*sistema operacional baseado em Linux*) desenvolvido por Crispin Cowan em 1998 com o nome de *SubDomain*, pouco tempo depois, especificamente em 2005 a empresa de softwares para redes e Internet "Novell" adquiriu a equipe desenvolvedora do

Immunix, e consequentemente do AppArmor, nisso o AppArmor começou a ser desenvolvido pela SUSE, que na época já havia sido comprada pela Novell.

O desenvolvimento do AppArmor ficou concentrado pela SUSE até 2009, quando uma parte dos direitos foi concedido a Canonical Ltda. que continua até hoje como sucessora oficial, principal fonte de suporte e desenvolvimento do AppArmor. Atualmente o AppArmor é integrado em múltiplas distribuições como um LSM devido a sua facilidade de configuração e estabilidade, contudo ele começou a ser parcialmente integrado a partir do Linux *kernel* 2.6.36, desde 20 de Outubro de 2010, onde a Canonical já estava responsável pelo AppArmor.

O AppArmor funciona através do confinamento de processos/programas/serviços em arquivos, *capabilities*, acessos de rede e *rlimits*, onde:

- Arquivos: demonstra os arquivos que será executados dentro do perfil do processo determinado do AppArmor
- *Capabilities*: *capabilities* é a representação de privilégios personalizados definidos pelo administrador para a execução do processo determinado na perfil, por exemplo "*capability setpcap allow*", onde isso permite que o serviço utilize as portas reservadas para se inicializar, portas tais como abaixo de 1024 tanto UDP quanto TCP, onde o processo "*setpcap"* teria tal permissão
- Rede: demonstra os privilégios de rede que o processo pode abrir dentro do perfil, desde receber até conectar em diferentes IPs, domínios etc…
- *Rlimits*: *rlimits* é o onde é definido o uso de recursos do sistema operacional pelo serviço determinado, por exemplo: "*rlimit cpu time 60*", onde isso delimita que o uso máximo de CPU para esse processo é de 60 segundos ou menos.

Devido à flexibilidade e sua abordagem *user-friendly* os administradores conseguem implementar os perfis sem a necessidade de reiniciar o sistema.

Quando o sistema operacional iniciar um processo ele será verificado primeiramente se há um perfil do AppArmor atrelado ao processo, caso um perfil exista o AppArmor aplicará as restrições e permissões respectivas do perfil carregado ao processo em execução. Caso o processo não possua um perfil atrelado ele não será restringido pelo AppArmor e suas permissões e restrições irão ser responsabilidade do permissionamento *default* do Linux por trás, ou seja, será responsabilidade do modelo DAC implementado por *default* como dito pela própria documentação do AppArmor.

> "*Programas não confinados são executados sob padrão Linux Discritionary Access Segurança de controle (DAC). O AppArmor aumenta o DAC tradicional nesses programas, os programas confinados são avaliados sob DAC tradicional primeiro e se o DAC permite a ação após isso o programa será então consultado pela política do AppArmor. (Steven Beattie; 2018; GettingStarted; p.1)*"

Existem duas ações que podem ser feitas na associação de um perfil para um processo, onde ela pode ser de *allow* ou *deny*, onde dentro do arquivo de perfil pode ser realizadas outras duas mudanças no funcionamento do processo que está a ser executado, caso o perfil possua a *flag "complain"* o processo que está confinado dentro do ambiente do AppArmor será executado e sua execução será feita um log de informações sobre. Caso o perfil possua a *flag* de "enforce" o processo será interrompido.

Figura 9: Representação de um perfil de processo no AppArmor

```
#include <tunables/global>
/bin/ping flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/consoles>
  #include <abstractions/nameservice>

  capability net_raw,
  capability setuid,
  network inet raw,

  /bin/ping mixr,
  /etc/modules.conf r,
}
```

Fonte: https://ubuntu.com/server/docs/security-apparmor

A figura 9 mostra como seria um perfil para um processo que acessa rede, neste caso o */bin/ping*, onde tal perfil pode ser dividido em algumas partes para explicação:

#includes <tunables/global>: permite que seja referenciado outras declarações feitas por outros serviços

/bin/ping flags=(complain): caminho para o programa, neste caso o *flags=(complain)* representa a ação que será feita pelo AppArmor caso o processo seja executado

capability: permite ao processo alguns privilégios, neste caso net_raw representa a permissão de acesso para CAP_NET_RAW, que basicamente permite um processo criar e se alocar a um *socket* de rede.

/bin/ping mixr:

Vale a pena notar que esse perfil representativo do */bin/ping* seria salvo em */etc/apparmor.d/* como o nome de */etc/apparmor.d/bin.ping*

Um administrador pode criar um perfil para qualquer processo que será executado dentro do sistema operacional, a única coisa necessária para ele saber será o funcionamento do próprio processo e que após a criação do perfil e sua associação do processo será necessário a reinicialização do AppArmor e do processo cujo perfil foi criado.

Uma funcionalidade interessante para o caso citado acima é se um administrador possui dúvidas sobre o funcionamento de tal processo, neste caso ele poderia criar um perfil para aprendizado (chamado de *per-profile learning*), onde o processo seria executado em um ambiente controlado pelo AppArmor e ele realizaria um *log* de quais permissões estariam sendo utilizadas, tais informações ajudariam na montagem de um perfil e uma política para a utilização desse perfil. Esse é o único caso de uso para a utilização de perfis determinados como *complain* para o AppArmor.

> "*O AppArmor oferece suporte ao modo de aprendizado per-profile(complain) para ajudar os usuários a escrever e manter a política. O modo de aprendizagem permite que um perfil seja criado executando um programa normalmente e aprendendo seu comportamento. Depois AppArmor aprendeu suficientemente o comportamento, o perfil pode ser colocado em modo de enforce. Embora o perfil resultante possa ser mais tolerante do que um perfil feito à mão sob medida para um determinado ambiente e aplicação, o modo de aprendizagem pode reduzir significativamente o esforço e conhecimento necessários para usar o AppArmor e adicionar um importante camada de segurança. (Steven Beattie; 2018; GettingStarted; p.1)*"

Como já foi dito existem vários mecanismos de controle de acesso, entre eles existe o MAC e o jeito mais comum de implementar o MAC em sistemas UNIX é através do SELinux.

O SELinux inicialmente foi desenvolvido pela empresa norte-americana de defesa nacional (NSA - *National Security Agency*) com o objetivo de melhorar o controle de servidores UNIX sem a necessidade de perder sua flexibilidade.

Segundo Stephen Smalley, 2005, houve alguns protótipos antes do lançamento do SELinux, porem eles eram para outros sistemas operacionais, entre eles são:

DTMach - *Distributed Trusted Mach*

Framework de implementação de MAC para o Mach Kernel

DTOS - *Distributed Trusted Operating System*

Versão melhorada o DTMach, com alguns incrementos no Mach Kernel

Flask - *Flux Advanced Security Kernel*

Implementação do DTOS para o Flask Kernel

Contudo não havia um projeto para a implementação no Linux em si, e os protótipos anteriores não possuíam suporte para os módulos de segurança integrados do Linux (LSM -

Linux Security Module), inicialmente o principal suporte e apoio para a criação desse módulo veio através da NSA e Linus Torvald.

Inicialmente o SELinux utilizava IDs persistentes de segurança para cada arquivo presente no sistema operacional, esses IDs eram valores numéricos de difícil interpretação para o usuário, contudo nisso surgiu 2 problemas na época com essa versão de SELinux, a primeira é que os IDs persistentes poderiam confundir os administradores de sistema ao gerenciá-los, e o segundo problema é que caso o administrador quisesse instalar o SELinux em um sistema operacional já em execução ele teria que acessar e modificar cada arquivo individualmente e atribuir os IDs manualmente, o que evidentemente, é um grande problema independente da escala do sistema operacional.

Nisso veio outra versão de SELinux para a sua implementação em *kernel* 2.4.x de Linux, contudo também não foi um grande sucesso devido a problemas de performance e havia problemas de inconsistência em múltiplas versões, contudo a partir do *kernel* 2.6.x o SELinux foi integrado com sucesso e também permitiu compatibilidade com o sistema de arquivos ext3. Atualmente a principal fonte de suporte e apoio de uso, desenvolvimento e suporte do SELinux vem diretamente da *NSA*, de desenvolvedores autônomos do SELinux e da *Red Hat*. Hoje em dia o SELinux implementa o MAC através de um modelo de permissionamento chamado RBAC (*Role-Based Access Control*)

Ao criar uma regra, o administrador de sistema deve pensar no seguinte questionamento:

> “Pode o SUJEITO fazer AÇÃO para aquele OBJETO’, por exemplo: poderia um servidor *web* acessar arquivos de usuários em seus *home directory?*” (Jahoda; Ančincová; Gkioka; Čapek; 2019 ; P.5)

Baseado nisso é necessário que o SELinux consiga definir o que cada processo/sujeito possa realizar ou não, a primeira parte desse processo é a atribuição de *tags,* ou como *SELinux context* como são comumente chamadas. Esse processo de classificação e padronização permite que um administrador de sistema um nível de gerenciamento muito mais granular sem necessariamente atribuir uma redundância ou ambiguidade para os processos, onde um processo pode ter links simbólicos ou múltiplos caminhos de acesso (*paths*). Ao criar uma regra a permissão é de negar por padrão, caso uma regra de somente acessar seja feita para um arquivo e não houver outras regras complementando as permissões, elas serão negadas.

Uma regra estabelecida pelo SELinux funciona através de vários contextos determinados pelo administrador do sistema, os contextos mais comuns são ‘usuários’ ,’cargo’, ’tipo’, e 'nível de segurança’, o administrador de sistema deve conhecer por completo o funcionamento dos serviços em execução de seu sistema operacional para assim conseguir criar regras que se condizem com o funcionamento normal do programa, caso contrário uma regra que

intuitivamente parece ser ‘OK’ na verdade está prejudicando o funcionamento normal do serviço, podendo causar instabilidades e problemas de integridade.

Sobre o contexto de ‘usuário’ das regras do SELinux, ele representa o usuário dentro do SELinux, normalmente os usuários do Linux são os mesmos do SELinux, contudo esses 2 “ambientes” são diferentes, isso permite que os usuários do ambiente Linux herdem as restrições implementadas pelo SELinux. Qualquer usuário pode ver as permissões aplicadas ao seu usuário.

O contexto de cargo é a aplicação definitiva do RBAC, o cargo define quais "domínios" podem ser acessados. A definição de domínio pode ser definida como os contextos dos objetos definidos no sistema operacional, ou seja, um domínio é basicamente os arquivos que definem a execução de um processo ou serviço, como: o serviço *MySQL* seria um domínio os arquivos em /var/lib/mysql, por exemplo, fariam parte desse domínio.

Tipo representa o tipo de ação que a regra está permitindo ao domínio e para quem, ou seja, se a ação é de permitir acesso de um domínio para outro, ou de um domínio para um arquivo de outro domínio, ou de um arquivo de um domínio para outro domínio etc…

Antes de explicar o que significa o level no SELinux é necessario explicar um modelo chamado MLS (*Multi-Level Security*) e outro chamado *Bell-La Padula Mandatory Access Control.*

A figura 10 demonstra o conceito básico de funcionamento do modelo *Bell-La Padula* foi criado por David E Bell e Leonard J.La Padulla com o objetivo de auxiliar o Departamento de Defesa Americano (DoD - *Departament of Defense*) a implementar um sistema que instalar por definitivo um modelo MAC e DAC na infraestrutura de TI do DoD, por padrão esse modelo tem o foco na confidencialidade da informação, por exemplo: uma informação sobre empresa pode variar desde Pública/Não Classificada, Confidencial, Secreta, ou Super Secreta, esse padrão de classificação foi feito pelo departamento de defesa dos Estados Unidos.

Contudo havia um problema nesse modelo que ele apenas focava em confidencialidade, logo se algum arquivo fosse alterado sem permissão ele não “era responsavel” por sua verificação de integridade.

Figura 10: Representação do modelo Bell-La Padula

Fonte: https://danieldonda.com/modelo-de-seguranca-bell-lapadula/

O modelo MLS é praticamente a implementação do *Bell-La Padula* contudo estabelecido como um "nível" das verificações realizadas pelos módulos de segurança implementados.

O nível de segurança, ou comumente chamado de apenas *level,* diz respeito a sensibilidade e categoria da informação, ou seja, 1 *level* de segurança do SELinux é composto por uma classificação de sensibilidade e sua categoria, exemplo: um documento interno é de categoria interna, logo não pode ser compartilhado publicamente, contudo é de sensibilidade desclassificada que significa que todos os internos podem ver a informação. O SELinux utiliza uma abordagem parecida, ele atribui a categoria da informação como “s0”, onde isso quer dizer que a sensibilidade deste recurso é o mais baixo possível, sendo de acesso público para todos no sistema operacional. Os valores *default* (máximo e mínimo) para a definição de informação é de “s0” até “s15”, contudo isso não é fixo e pode ser modificado pelo administrador do sistema para adicionar mais flexibilidade.

Essa representação de “s0,s1,sn” é a implementação do MLS no SELinux, contudo ele mais mecanismos permitir ou negar, outro entre eles é o MCS, onde similarmente é representado como “c0,c1,c2” até um “c1023”, podendo serem aplicadas textos para cada classificação para ajudar na definição do nível. Ele é aplicado em processos e arquivos e só podem ser acessados, escritos e executados por outros usuários/processos que possuam os

mesmos níveis, logo um processo que contenha “c0.c5” (c0,c1,c2,c3,c4,c5) só pode ser executado por um usuário que tenha os mesmos 6 níveis. Aqui o “c” representa a categoria.
Outro conceito importante sobre o SELinux é o conceito de Confinamento, onde existe dois tipos de casos que podem acontecer com os processos:

- **Processos confinados**
- **Processos não confinados**

Os processos confinados são processo que são executados dentro do seu próprio domínio, ou seja, ele só poderá executar os arquivos dentros do seu domínio, exemplo: caso o serviço do apache server esteja confinado e um invasor consiga comprometê-lo com uma vulnerabilidade como uma *webshell*, o serviço do apache (httpd) só poderá acessar, modificar ou executar os arquivos que estiverem dentro do seu domínio (httpd_t), tornando o impacto da vulnerabilidade muito menor e quase impossibilitando a pós exploração.
Por padrão todos os serviços que interagem com rede, usuários ou são executados como root são confinados, e os outros não são confinados, contudo o administrador do sistema pode mudar isso a qualquer hora.

Os serviços não confinados são executados em um domínio padrão para todos, o “*unconfined_service_t”* ou “*kernel_t”* dependendo se a sua execução seja realizada pelo *init* ou pelo *kernel* respectivamente, esses serviços não-confinados não são protegidos pelo SELinux e são somente restringidos pelas regras do DAC já implementadas no sistema, usando o mesmo exemplo de serviços confinados, caso o servidor apache seja comprometido e uma *webshell* seja executada, o atacante poderá realizar quaisquer ações dentro das restrições implementadas pelo DAC, como criar arquivos em */tmp,* algo que não seria possível se o processo estivesse confinado.
Importante notar que o SELinux e qualquer outra arquitetura MAC não é uma substituição da arquitetura DAC e sim um complemento, tanto que as regras do SELinux somente serão executadas após a aprovação das regras impostas pelo DAC, caso elas sejam negadas no DAC elas não serão nem executadas pelo SELinux, como dito pelo livro da Red Hat *SELinux User’s and administration’s Guide.*

> *“As regras DAC ainda são usadas. O SELinux é um aprimoramento de segurança sobre as regras DAC, ele não as substitui.*
> (Jahoda; Ančincová; Gkioka; Čapek; 2019 ; P.21)”
> “*As regras de política SELinux são verificadas após as regras DAC. As regras de política SELinux não são usadas se as regras DAC negarem o acesso primeiro.*
> (Jahoda; Ančincová; Gkioka; Čapek; 2019 ; P.38)”

7. Conclusão

Por fim este trabalho teve o objetivo de explicar os principais motivos para que um administrador monitore os serviços de seu sistema operacional, demonstrando ferramentas que podem auxiliar as atividades de verificação de integridade dos serviços e automatizar suas tarefas diárias, alem disso é reforçado a necessidade de manter os serviços atualizados e uma verificação constante no permissionamento dos usuários presentes no sistema operacional. Também é demonstrado a importância de conhecer o ambiente de execução de tal sistema operacional para que, caso seja necessário, seja adotado um novo modelo de permissionamento especializado para o ambiente, por exemplo: em ambientes governamentais, saúde, militares e de alta tecnologia é comum encontrar modelos mais restritivos de permissionamento como o MAC, enquanto em sistemas mais toleráveis a permissão de seus usuários e suas informações utilizam o DAC.

Outro ponto importante que é demonstrado é que a comunidade de segurança contribui ativamente para o desenvolvimento de ferramentas que auxiliam os administradores em suas tarefas, um exemplo disso é a ferramenta/site VirusTotal [8]que foi demonstrada em auxiliar na detecção de vírus em arquivos, independente do tipo.

É explicado que o desenvolvimento desses modelos não surgiu do nada, e sim o produto de pesquisa para implementação de modelos e ferramentas de segurança em sistemas de alta criticidade, por exemplo: no caso do SELinux, foi a necessidade de adaptar modelos e ferramentas que implementavam o MAC em outros *kernels* para os *kernels* UNIX.

Contudo existem outras possibilidades que também devem ser abordadas por administradores que exigem a atenção dos administradores, por exemplo: onde *updates* automáticos podem ocasionalmente introduzir novas vulnerabilidades nos sistemas operacionais, onde podemos citar dois casos:

- PHP 8.1-dev[9]: Um *update* em 28 de Maio de 2021 continha um backdoor que permitia que qualquer atacante executasse código remoto não autenticado no servidor alvo
- *SolarWinds Orion Hack* [10] : Agentes maliciosos conseguiram acesso a infraestrutura da *SolarWinds* e injetaram código malicioso em um programa chamado *Orion*, que realizava monitoramento de rede em alguns clientes governamentais e privados ao redor do mundo, após testar o código malicioso

[8] https://www.virustotal.com/gui/home/

[9] https://www.exploit-db.com/exploits/49933

[10] https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know

os *hackers* injetaram esse código em no sistema de *updates* do *Orion* e todos os próximos *updates* são lançados com código malicioso

Porém, por mais que novos *updates* possam ser problemáticos, sendo que alguns *updates* automaticos podem causar problemas de disponibilidade e conflitos no sistema operacional, ainda é recomendado atualizar os seus serviços para previnir que vulnerabilidades de versões desatualizadas tenham efeito, no final é uma questão de probabilidade, onde é mais provável ter vulnerabilidades em versões desatualizadas do que em novas versões.

Essa área pode ser inspirar futuras pesquisas, onde o autor pode demonstrar outros casos de *updates* que causaram problemas e até mesmo desenvolver técnicas e ferramentas para detectar *updates* maliciosos e filtra-los.

8. Referencias

BAIER, Danny, A Brief History of RBAC and ABAC, 2022, Disponivel em: https://www.styra.com/blog/a-brief-history-of-rbac-and-abac/

BREUKER, Dominic; Pspy - unprivileged Linux process snooping, Disponivel em https://github.com/DominicBreuker/pspy

CentOS, SELinux, 2020, Disponivel em: https://wiki.centos.org/HowTos/SELinux
SUSE, Security Guide, 2023, Disponivel em: https://documentation.suse.com/sle-micro/5.1/html/SLE-Micro-all/article-selinux.html

DONDA, Daniel. Modelo de Segurança : Bell–LaPadula, 2021, Disponivel em: https://danieldonda.com/modelo-de-seguranca-bell-lapadula/

DAVIS, Christopher; MAC vs DAC vs RBAC, 2014, Disponivel em: http://www.cloudauditcontrols.com/2014/09/mac-vs-dac-vs-rbac.html

FLAST101, PHP 8.1.0-dev - 'User-Agentt' Remote Code Execution , Disponivel em: https://www.exploit-db.com/exploits/49933

FRANCO, Luis; SAHAMA, Tony; CROLL, Peter; Security Enhanced Linux to Enforce Mandatory Access Control in Health Information Systems, 2008, Disponivel em: https://www.researchgate.net/profile/Peter-Croll/publication/41201865_Security_Enhanced_Linux_to_enforce_Mandatory_Access_Control_in_Health_Information_Systems/links/02bfe5108d1cfce9b2000000/Security-Enhanced-Linux-to-enforce-Mandatory-Access-Control-in-Health-Information-Systems.pdf

HERTZOG, Raphaël; MAS, Roland. The Debian Administrator's Handbook. Freexian SARL, v. 16, 2013, Disponivel em <https://debian-handbook.info/download/squeeze/debian-handbook.pdf>

ISC², 2023, Disponivel em: https://www.coursera.org/lecture/access-control-sscp/implement-access-controls-mandatory-access-control-3qT8Z

JAHODA, Mirek; ANČINCOVÁ, Barbora; GKIOKA, Ioanna; ČAPEK, Tomáš, SELinux User's and Administrator's Guide, 2019, Disponivel em: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/pdf/selinux_users_and_administrators_guide/red_hat_enterprise_linux-7-selinux_users_and_administrators_guide-en-us.pdf

KERNER, Sean; OLADIMEJI ,Saheed; Disponivel em: https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know

LI, Vickie; Becoming Root Via a Misconfigured PATH 2020, Disponivel em: https://betterprogramming.pub/becoming-root-via-a-misconfigured-path-720a52981c93

NEMETH, Evi; SNYDER, Garth; HEIN, Trent R.; WHALEY, Ben; MACK, Dan Unix and Linux Administration Handbook, Addinson-Wesley, 2018, Disponivel em <https://mog.dog/files/SP2019/2017%20Nemeth%20Evi%20etal%20-%20UNIX%20and%20Linux%20System%20Administration%20Handbook%5B5thED%5D_Rell.pdf>

PENG, Peng; YANG, Limin; SONG, Linhai, WANG, Gang, Opening the Blackbox of VirusTotal: Analyzing Online Phishing Scan Engines, 2019, Disponivel em https://dl.acm.org/doi/pdf/10.1145/3355369.3355585

REDAÇÃO, Debian, Syslog man pages, Disponivel em : https://manpages.debian.org/jessie/manpages-dev/syslog.3.en.html

REDAÇÃO, Debian, Cron man pages , Disponivel em: https://manpages.debian.org/unstable/cron/cron.8.en.html

REDAÇÃO, Die.net, Logcheck man pages, Disponivel em https://linux.die.net/man/8/logcheck

REDAÇÃO, Kaspersky; What is data theft and how to prevent it, 2018, Disponivel em: https://www.kaspersky.com/resource-center/threats/data-theft

CASEY, Eoghan; STELLATOS, Gerasimos; The Impact of Full Disk Encryption on Digital Forensics, 2008.

REDAÇÃO, Kingston; What is SSD encryption and how does it work?, 2021 Disponivel em https://www.kingston.com/en/blog/data-security/how-ssd-encryption-works

REDAÇÃO, Ubuntu, Top man pages, Disponivel em: https://manpages.ubuntu.com/manpages/xenial/man1/top.1.html

Red Hat, Endurecimento da segurança, 2021 Disponivel em: https://access.redhat.com/documentation/pt-br/red_hat_enterprise_linux/8/html/security_hardening/encrypting-block-devices-using-luks_security-hardening

Red Hat; Using SELinux, 2023, Disponivel em: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/using_selinux/using-multi-level-security-mls_using-selinux

SMALLEY, Stephen; Integrating Flexible Support for Security Policies into the Linux Operating System, 2005, Disponivel em: https://www.nsa.gov/portals/75/documents/what-we-do/research/selinux/documentation/presentations/2005-flexible-support-for-security-policies-into-linux-os-presentation.pdf

Akito, Stack Exchange, List available methods of encryption for LUKS, 2019, Disponivel em https://unix.stackexchange.com/questions/354787/list-available-methods-of-encryption-for-luks