**TEMAS**

======================== Processo de Forense no Linux ======================== Como coletar dados para iniciar uma investigação e perícia forense em ambientes Linux ------------------------

# INTRO

Aqui veremos algumas ferramentas que podemos utilizar para fazer um processo de análise forense em máquinas Linux (ou seja, boa parte dos sistemas presentes em servidores ao redor do mundo). Para quem está caindo de para quedas aqui, um processo de análise forense computacional é instaurado (ou iniciado) quando ocorrem incidentes em ambientes que comproteram elementos da segurança que podem ser de ordem de Confidencialidade, Integridade ou Disponibilidade do sistema. O motivo da perícia é descobrir como o incidete ocorreu para reconstruir os passos do atacante, descobir qual foi o agente deste incidente e aprender como tudo ocorreu para estabelecer melhores defesas contra aquele caso. Claro que uma perícia forense não se resume apenas a isso, mas isso serve de introdução para quem é totalmente leigo no assunto.

Vale ressaltar que antes de utilizar algumas ferramentas para coleta de informações do sistema é de extrema importância configurações prévias pois alguns dados podem não ser coletados após um incidente por diversos motivos como: logs não são armazenados no sistema operacional automaticamente ao menos que preparemos o sistema para isso, nem todas informações são armezanadas nesses logs e precisamos configurar o sistema de auditoria do sistema, tais informações podem ser removidos do sistema por atacantes ou outros elementos que possam inferir nos dados do sistema portanto cabe a centralização dos logs do sistema em servidores externos que estejam responsáveis pelo processo de armazenameto e categorização desses dados de logs do sistema, ferramentas como Grafana e Zabbix podem ser eficientes neste caso. Como este não é o objetivo deste artigo vamos nos ater apenas a análise no ambiente Linux puro, mantendo a analise de forma centralizada através dessas ferrmentas para outros estudos. Agora, vamos começar o processo de aprendizado em ferramentas forenses para servidores Linux? Precisaremos coletar e compilar o máximo de informações sobre o sistema que precisa ser periciado. Para isso teremos alguns passos a seguir que trarão iformações diferentes.

LEVANTAMENTO DE INFORMAÇÕES ##########################

Primeiro vamos levantar os detalhes de nosso sistema operacional com o comando uname -a. Com este commando vamos extrair alguns detalhes sobre o sistema que são cruciais, vamos analisar o retorno em um caso típico:

Linux PcName 5.19.0-000-generic #202212242330 SMP PREEMPT\_DYNAMIC Mon Jul 15 16:40:02 UTC 2022 x86\_64 x86\_64 x86\_64 GNU/Linux

* **Linux** - Tipo do sistema
* **PcName** - Nome de Host
* **5.19.0-000-generic** - Versão do Kernel, composto por: Versão-Release-Tipo
* **#202212242330** - Numero de Compilação do Kernel
* **SMP** - Indicador de suporte do Kernel a multi processadores e nucleos
* **PREEMPT\_DYNAMIC** - Indicador de suporte a preempção dinâmica
* **Mon Jul 15 16:40:02 UTC 2022** - Data e hora em que o Kernel foi compilado
* **x86\_64 x86\_64 x86\_64** - Indicador da Arquitetura do Sistema
* **GNU/Linux** - Indicador de distibuição baseada no GNU

Depois o comando será hostnamectl, embora o output do mesmo seja bem intuitivo, vamos analisar com mais ímpeto o seu retorno no sistema:

Static hostname: PcName  
 Icon name: computer-laptop  
 Chassis: laptop  
 Machine ID: ec8a47fe4a75fe65a4ef76505505fb80  
 Boot ID: 6009e87f78e9f7e9f7e9eaedc89e0d38  
Operating System: Zorin OS 17.1   
 Kernel: Linux 6.5.0-45-generic  
 Architecture: x86-64  
 Hardware Vendor: Positivo Bahia - VAIO  
 Hardware Model: VAIO XPTO

* **Static hostname** - identificação do sistema em uma rede e resolução de nomes de domínio.
* **Icon name** - Representa qual o tipo de hardware.
* **Chassis** - Descreve o tipo físico do dispositivo.
* **Machine ID** - Identificador único da máquina, que é gerado automaticamente. Crucial para a identificação única do sistema, ótimo para ambientes gerenciados centralmente, como no gerenciamento, configuração e orquestração de sistemas.
* **Boot ID** - Identifica a sessão de boot atual, sendo assimm útil para diagnóstico e auditoria do sistema, rastreando eventos específicos de sessões de boot.
* **Operating System** - O nome do sistema operacional instalado.
* **Kernel** - A versão do kernel Linux em uso.
* **Architecture** - A arquitetura da CPU (e.g., x86\_64, arm).
* **Hardware Vendor** - O fabricante do hardware.
* **Hardware Model** - O modelo específico do hardware.

Continuando, agora vamos usar o comando cat /etc/os-release, que trará apenas mais informmações sobe o hostame, com isso teremos o output abaixo:

PRETTY\_NAME="Zorin OS 17.1"  
NAME="Zorin OS"  
VERSION\_ID="17"  
VERSION="17.1"  
VERSION\_CODENAME=jammy  
ID=zorin  
ID\_LIKE="ubuntu debian"  
HOME\_URL="https://zorin.com/os/"  
SUPPORT\_URL="https://help.zorin.com/"  
BUG\_REPORT\_URL="https://zorin.com/os/feedback/"  
PRIVACY\_POLICY\_URL="https://zorin.com/legal/privacy/"  
UBUNTU\_CODENAME=jammy

Outros processos importantes a serem levantados no sistema operacional são os dados de BIOS, processador e memória. Para levantamento de dados do BIOS podemos usar o comando sudo dmidecode onde podemos coletar dados como Vendor, Versão, Data de Lançamento, Endereço na Memória, Características, Nome do Produto de acordo com o fabricante, numero de série entre outras informações que auxiliam na identificação do dispositivo.

# USERS

Agora vamos começar a coletar informações dos usuários do sistema para catalogar suas permissões e dados informativos, para tal começaremos com o comando w, sim, apenas a letra "w" mesmo ou, caso queira uma versão mais enxuta do output pode usar o camando who. Neste comando encontraremos o output:

20:33:01 up 2:30, 3 users, load average: 0.02, 0.03, 0.00  
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT  
user1 pts/0 192.168.0.101 20:00 1:30 0.02s 0.02s -bash  
user2 pts/1 192.168.0.102 20:10 2:00 0.01s 0.01s -bash  
user3 pts/2 192.168.0.103 20:20 0.00s 0.00s 0.00s w

* **20:33:01** - Sendo o horário atual.
* **up 2:30** - Sendo o tempo de atividade do sistema (uptime).
* **3 users** - Que é o nNúmero de usuários logados.
* **load average: 0.02, 0.03, 0.00** - Média de carga do sistema nos últimos 1, 5 e 15 minutos.
* **USER** - O nome do usuário.
* **TTY** - O terminal ao qual o usuário está conectado.
* **FROM** O endereço IP ou hostname de onde o usuário está logado.
* **LOGIN@** - A hora em que o usuário fez login.
* **IDLE** - Tempo de inatividade do usuário.
* **JCPU** - Tempo total de CPU utilizado por todos os processos associados ao terminal.
* **PCPU** - Tempo de CPU utilizado pelo processo em execução atualmente.
* **WHAT** - O comando ou processo que o usuário está executando atualmente.

Para auditarmos tentativas de acesso do usuário e quantos processos estão rodando podemos usar o comando sudo lslogins, onde teremos o resultado a seguir:

UID USER PROC PWD-LOCK PWD-DENY LAST-LOGIN GECOS  
 0 root 156 0 1 root  
 1 daemon 0 0 1 daemon  
 2 bin 0 0 1 bin  
 3 sys 0 0 1 sys  
 4 sync 0 0 1 sync  
 5 games 0 0 1 games  
 6 man 0 0 1 man  
 7 lp 0 0 1 lp  
 8 mail 0 0 1 mail  
 9 news 0 0 1 news  
 10 uucp 0 0 1 uucp

* **UID** - User Identifier (Identificador do Usuário)
* **USER** - Nome do usuário
* **PROC** - Número de processos pertencentes ao usuário e atualmente em execução
* **PWD-LOCK** - Indica se a conta do usuário está bloqueada
* **PWD-DENY** - Indica se o login por senha está negado
* **LAST-LOGIN** - Data e hora do último login
* **GECOS** - Informações adicionais sobre o usuário como o nome completo, telefone e detalhes de contato.om "root".

Um comando que pode auxiliar neste processo é o sudo finger que trará mais informações do usuário atual, apresentanddados que podem ser utilizados para compreender o horario do ultimo login e terminais acessados;

Login Name Tty Idle Login Time Office Office Phone  
efraim Efraim \* :0 Jul 29 09:12 (:0)  
efraim Efraim pts/2 Jul 29 20:39

* **Login** - Nome de login do usuário.
* **Name** - Nome completo do usuário.
* **Tty** - Terminal ou pseudo-terminal em uso.
* **Idle** - Tempo de inatividade do usuário.
* **Login Time** - Data e hora do login.
* **Office** - Informação de escritório do usuário (estará em /etc/passwd)
* **Office Phone** - Número de telefone do escritório.

# BEHAVIOR

Para termos maior acurácia neste momento, precisamos ter ferramentar para detectar o comportamento dos usuários no sistema (que podem até ser aduterados, por isso existe todo um processo a ser seguido antes que uma perícia seja intaurada).

Os comandos abaixo levantarão o comportamento do sistema atrelado a dados como IPs e usernames para que possamos triangular as ações efetuadas no sistema.

Um comando que gosto de usar para auditar eventos no sistema é o last -Fxiw, onde encontraria output semelhante a este:

runlevel (to lvl 5) 0.0.0.0 Fri Jun 28 22:14:29 2024 - Sat Jun 29 21:40:01 2024 (23:25)  
reboot system boot 0.0.0.0 Fri Jun 28 22:14:18 2024 - Sat Jun 29 21:40:01 2024 (23:25)

Aqui vamos separar por coluna para uma melhor compreensão de cada evento (linha):

* **Tipo de Evento** - podemos ter diversos tipos de eventos, vale a pena começar a analise por este campo.
* **Informações Adicionais** - uma breve descrição do ocorrido.
* **IP ou hostname** - Apresenta o IP de origem da sessão, eventos em IP 0.0.0.0 são representam um evento local.
* **Data** - Apresenta data e hora do início e horário do evento ou sessão, após o héfen "-" é apresentado a data e hora finais.
* **Período** - O período em que este evento durou (no formto hh:mm).

Agora, para auditarmos o comportamento do usuário de forma mais acurada ainda podemos utilizar o comando cat /var/log/auth.log que pode ser ainda mais potencializado em conjunto com o "grep" para detectarmos ações específicas em meio aos logs, assim como grep -a sudo /var/log/auth.log que retornará:

Jul 29 21:54:30 zorin sudo: pam\_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)  
Jul 29 21:54:30 zorin sudo: pam\_unix(sudo:session): session closed for user root

Aqui encontraremos os seguintes elementos: \* **Data** - data e hora em que o evento ocorreu \* **Hostname** - o nome do host em que o evento ocorreu \* **Origem do Log** - serviço ou comando em que o comando está relacionado \* **Auth Module** - é o modulo de autenticação responsável pelo evento \* **Tipo de Evento** - seria como um campo de detalhes do evento

Tendo uma idéia do comportamento geral do usuário podemos agora buscar por pistas de seu comportamento perante execuções no sistema operacional, isso significa que iremos auditar suas ações em função de sessões de boot do sistema. Para tal, utilizaremos comandos como ps -p e systemd-cgls mas em função de uma sessão específica do boot que poderia ser a do dia em que o incidente ocorreu ou então de um dia compreendido em um range de datas cujo o incidente possa ter ocorrido. Vamos partir da premissa que precisamos analisar o que está ocorrendo em tempo real na nossa máquina Linux, isso é extremamente importante para que possamos diferenciar processos legítimos de possíveis processos maliciosos no sistema operacional. Para termos uma boa visão de todos os processos executados e analisar o que pode estar ocorrendo podemos utilizar o comando ps -eFH

Para conseguirmos as datas e os Boot ID correspondentes a essas datas usaremos os comandos journalctl --list-boots, note que para ter efetividade neste comando é necessário ter iniciado a captura de Boot IDs e a persistência dos logs no sistema no sistema, é muito importante que faça o processo de persistência de Logs no sistema seja configurado no início da utilização do sistema operacional, uma dica seria criar um shell script de inicialização, mas para fazer isso manualmente basta editar o arquivo "/etc/systemd/journald.conf" através do parâmetro mencionado abaixo:

[Journal]  
Storage=persistent

Caso esta linha esteja comentada, descomente-a e caso não exista, crie.

Uma abordagem muito comum em instalação

cat /etc/crontab cat /etc/cron.hourly/ cat /etc/cron.daily/ cat /etc/cron.weekly/ cat /etc/cron.monthly/ cat /etc/cron.d/ crontab -u $USER -l cat /var/log/cron sudo grep cron /var/log/syslog

sudo bash -c ‘for user in $(cut -f1 -d: /etc/passwd); do entries=$(crontab -u $user -l 2>/dev/null | grep -v “^#”); if [ -n “$entries” ]; then echo “$user: Crontab entry found!”; echo “$entries”; echo; fi; done’ sudo ls -la /var/spool/cron/crontabs

sudo systemctl list-units — all — type=service Um comando que pode ser muito útil para descobrir serviços configurados no sistema que podem ser maliciosos é o comando systemctl list-unit-files, este comando retorna os arquivos de configuração dos serviços, isso pode facilitar nossa busca por serviços maliciosos, como sempre preciasamos fazer no Linux vale a pena analizar o resultado do output filtrando os resultados com "grep", para este caso vamos enfatizar alguns termos que podem nos dar retornos mains interessantes para nossos serviços, tais termos são: \* service \* device \* mount \* socket

O output deste comando nos retornará alguns dados que podem ser interpretados como:

* **UNIT FILE:** Nome do arquivo de unidade. Isso inclui o tipo de unidade (por exemplo, .service, .mount, .socket).
* **STATE:** Estado de habilitação da unidade. Os estados comuns incluem:
  + enabled: A unidade está habilitada e será iniciada automaticamente no boot.
  + disabled: A unidade está desabilitada e não será iniciada automaticamente.
  + static: A unidade não pode ser habilitada diretamente, pois é usada apenas como dependência de outras unidades ou é ativada por outra unidade.
  + masked: A unidade está mascarada, o que significa que ela está simbolicamente vinculada a /dev/null e não pode ser iniciada.

Outra abordagem que pode ser abordada para analisar os processos é através do diretório /proc, que organiza os processos de forma hierarquica; junto a isso podemos fazer uma analise dos processos e seus PIDs detectados para analisarmos o que cada processo tem feito no sistema através das ferramentas como top e htop para levantar qual seria o processo a ser analisado e seus PID. Com o PID em mãos podemos utilizar ferramentas como ps -f <PID>, lsof -p <PID> e pstree -p -s <PID>, assim teremos maior noção da ação que este processo está tomando no sistema operacional tanto em detalhes de recursos que o mesmo está acessando no background como de cadeia de processos que circundam o mesmo.

1. **Identificando a sessão de boot correspondente ao incidente:** Com a persistencia de logs agora conseguimos analisar os logs de acordo com o boot ID e seu timestamp, selecionando a data em que houve o incidente no sistema, para encontrar o boot ID já sabemos, basta usar o commando journalctl --boot-list e coletar o boot ID equivalente a data do incidennte (ou do evento que precisa analisar); tendo o Boot ID em mãos insira no comando journalctl -b <boot ID>, com isso terá acesso a todos os logs gerados naquele dia de forma completa. Mas mesmo tendo acesso aos logs pode ser difícil extrair dados no meio do volume de dados coletados, para isso podemmos usar comandos e prompts para facilitar na busca, segue abaixo alguns comandos que podem ser utilizados para encontrarmos dados que poderiam ser coerentes.

journalctl -b <boot\_id> | grep -i 'COMMAND'  
journalctl -b <boot\_id> | grep -i 'USER'  
journalctl -b <boot\_id> | grep -i 'net'  
journalctl -b <boot\_id> | grep -i 'dhcp'  
journalctl -b <boot\_id> | grep -i 'interface'  
journalctl -b <boot\_id> | grep -i 'iptables'  
journalctl -b <boot\_id> | grep -i 'socket'  
journalctl -b <boot\_id> | grep -i 'Started\|Stopped\|Enabled\|Disabled'  
journalctl -b <boot\_id> | grep -i 'modification\|changed\|updated'  
journalctl -b <boot\_id> | grep -i 'warning\|err\|critial\|alert\|emerg'  
journalctl -b <boot\_id> -p err  
  
#pequena pausa para informar que pra analisar serviços do sistema vale a pena ter uma idéia de quais são os serviços presentes no sistema  
systemctl list-unit-files --type=service  
journalctl -b <boot\_id> -u <service\_name>.service

Tendo uma boa noção dos comandos que podemos utilizar podemos agora emitir o comando e analisar sua saída para termos uma idéia de como os dados são coletados e como podem ser analizados. O comando utilizado para este caso foi o journalctl -b -3 | grep 'COMMAND'. Vamos analisar um pouco o output de tal comando:

ago 01 10:15:04 zorin pkexec[4875]: efraim: Executing command [USER=root] [TTY=unknown] [CWD=/home/efraim] [COMMAND=/usr/lib/update-notifier/package-system-locked]  
ago 01 10:23:46 zorin sudo[6980]: efraim : TTY=pts/2 ; PWD=/home/efraim ; USER=root ; COMMAND=/usr/bin/mkdir /tmp/teste  
ago 01 13:54:03 zorin pkexec[21772]: efraim: Executing command [USER=root] [TTY=unknown] [CWD=/home/efraim] [COMMAND=/usr/lib/update-notifier/package-system-locked]  
ago 01 20:38:37 zorin sudo[52261]: efraim : TTY=pts/0 ; PWD=/home/efraim ; USER=root ; COMMAND=/usr/bin/apt update  
ago 01 20:39:07 zorin pkexec[52798]: efraim: Executing command [USER=root] [TTY=unknown] [CWD=/home/efraim] [COMMAND=/usr/lib/update-notifier/package-system-locked]  
ago 01 20:39:23 zorin sudo[52809]: efraim : TTY=pts/0 ; PWD=/home/efraim ; USER=root ; COMMAND=/usr/bin/apt list --upgradable  
ago 01 20:39:45 zorin sudo[52819]: efraim : TTY=pts/0 ; PWD=/home/efraim ; USER=root ; COMMAND=/usr/bin/apt upgrade  
ago 01 22:16:05 zorin sudo[56218]: efraim : TTY=pts/0 ; PWD=/home/efraim ; USER=root ; COMMAND=/usr/sbin/shutdown now

* **dado** -
* **ago 01 10:12:10:** - É o timestamp, ou seja, permite identificar quando o evento ocorreu.
* **zorin:** - Hostname, para sabermos em qual host ocorreu o evento.
* **pkexec[4235]:** - Este é o comando ou programa que iniciou o evento, em outras literaturas já vi chamarem de facilities.
  + **[4235]:** - Aqui seria o ID do processo (PID) que gerou o log.
* **efraim::** - Este é o usuário que executou o comando.
* **Executing command:** - Aqui está o comando que foi executado.
* **[USER=root]:** - O usuário efetivo sob o qual o comando foi executado.
* **[TTY=unknown]:** - Representa o terminal (TTY) de onde o comando foi executado. No caso de pkexec, pode ser unknown porque o comando pode não estar ligado a um terminal específico.
* **[CWD=/home/efraim]:** - Representa o diretório de trabalho (CWD significa Current Working Directory) do usuário quando ele executou o comando.
* **[COMMAND=/usr/lib/update-notifier/package-system-locked]:** - Indica O comando que foi executado.

# DEPENDENCIES

Pode ser imprescindível analizar o quê está instalado no sistema, se os apps são legí timos ou podem estar comprometendo o sistema.

Para isso temos alguns comandos que podem ser emitidos no terminal para conseguirmos estes dados como informação.

Um primeiro comando que podemos emitir é o lsmod para verificarmos, na ordem do output, os módulos do Kernel, seu tamamho e quantidade em uso.

Module Size Used by  
vmnet 73728 17  
parport\_pc 53248 0  
vmmon 167936 0

Também precisamos analisar todos pacotes instalados, pode haver algo no meio, pra isso podemos usar comandos como dpkg -l, dpkg-query -l, apt list --installed, flatpak list e/ou snap list e verificar cada item e sua proveniência.