Ciberseguridad

Apuntes sobre auditorías, hardening, escaneos y herramientas que uso para asegurar sistemas sin volverme loco.

Auditorías de seguridad

Auditorías de seguridad

GitLeaks: Identifica secretos expuestos en repositorios de código

Alguna vez se cuela una API key en un commit y no apetece que acabe expuesta. Con **Gitleaks** se pueden detectar secretos antes de que sea demasiado tarde.

Instalación rápida

1. Descargar el binario desde Gitleaks Releases.
2. Dar permisos de ejecución:

```
chmod +x gitleaks
```

Listo para usar desde la terminal.

Escaneos básicos

Repositorio local

```
./gitleaks detect --source /ruta/del/repositorio
```

Repositorio remoto

```
./gitleaks detect --repo-url https://github.com/usuario/repositorio.git
```

Repositorio en Gitea

```
./gitleaks -v dir /Datos/Gitea/ChronosCMPS
```

Ejemplo de salida

```
Finding:     ND_LASTFM_APIKEY: "1234567890abcdef..."
RuleID:      generic-api-key
File:        /Navidrome/docker-compose.yml
Line:        23
...
```

En este caso detectó una API key dentro de un `docker-compose.yml`.

Ajustar la detección

- Guardar resultados en JSON:

```
./gitleaks detect --source /ruta --report-format json --report-path resultado.json
```

- Añadir reglas o excluir rutas según convenga.

Todas las opciones están disponibles con:

```
./gitleaks --help
```

Buenas prácticas

- Mantener Gitleaks actualizado.
- Integrarlo en pipelines de CI/CD.
- No ignorar alertas: una clave expuesta trae problemas.

Referencias

- Documentación oficial de Gitleaks

Bastionado de sistemas

Analizar servidores con NMAP

Nmap (Network Mapper) es una herramienta para comprobar el estado de redes y servidores. Aquí tienes una recopilación práctica de opciones y comandos que suelo usar, organizada como referencia rápida.

Conceptos básicos

- **Puertos abiertos**: Aceptan conexiones (ej. puerto 80 para HTTP).
- **Puertos cerrados**: No aceptan conexiones, pero existen.
- **Puertos filtrados**: No se sabe si están abiertos o cerrados; suelen estar protegidos por firewalls.
- **Rango de IPs**: Ejemplo: `192.168.1.0/24` cubre de `.1` a `.254`.

Especificación de objetivos

Opción	Descripción
`<host>`	IP, dominio o rango. Ej: `192.168.0.1-100`.
`-iL <archivo>`	Lista de objetivos desde archivo.
`-iR <n>`	Objetivos aleatorios.
`--exclude <host1[,host2]>`	Excluir hosts manualmente.
`--excludefile <archivo>`	Excluir desde archivo.

Descubrimiento de hosts

Comando	Descripción
`-sL`	Lista sin escanear.

Comando	Descripción
`-sn`	Solo ping scan.
`-Pn`	Trata todos los hosts como activos.
`-PS <puertos>`	Ping TCP por puertos.
`-PE`	Ping ICMP Echo.
`-PP`	ICMP timestamp.
`-PO <protocolos>`	Ping por protocolo.
`--traceroute`	Muestra la ruta al host.

Técnicas de escaneo

Comando	Descripción
`-sS`	SYN scan (rápido y silencioso).
`-sT`	TCP connect.
`-sA`	ACK scan (firewall mapping).
`-sU`	UDP scan.
`-sN`, `-sF`, `-sX`	Null, FIN y Xmas (evasión).
`-sO`	Protocolos IP.
`-sI <host>`	Idle scan con zombi.
`-b <host>`	FTP bounce.

Puertos y rangos

Comando	Descripción
`-p <rango>`	Ej: `-p 22,80`, `-p 1-1000`.
`-F`	Escaneo rápido de puertos comunes.
`--top-ports <n>`	Escanea los más usados.
`--exclude-ports <p>`	Excluye puertos.

Servicio y versión

Comando	Descripción
`-sV`	Detecta versión del servicio.
`--version-intensity`	Agresividad (0-9).
`--version-light`	Pruebas comunes.
`--version-all`	Pruebas completas.

Scripts NSE

Comando	Descripción
`-sC`	Scripts por defecto.
`--script <nombre>`	Scripts específicos.
`--script-args`	Argumentos para scripts.
`--script-trace`	Ver tráfico de los scripts.
`--script-updatedb`	Actualiza base de datos NSE.

Rendimiento y tiempo

Comando	Descripción
`-T0` a `-T5`	Velocidad de escaneo.
`--min-rate`	Paquetes por segundo mínimo.
`--max-rate`	Límite superior de velocidad.
`--host-timeout`	Tiempo máximo por host.

Evasión y spoofing

Comando	Descripción
`-f`	Fragmentar paquetes.
`-D <decoys>`	IPs señuelo.
`-S <IP>`	IP falsa como origen.
`--spoof-mac <mac>`	MAC falsa.
`--ttl <valor>`	TTL personalizado.

Formatos de salida

Comando	Descripción
`-oN`	Texto normal.
`-oX`	XML.
`-oG`	Grepeable.
`-oA`	Todos los anteriores.
`--reason`	Muestra el porqué del estado.
`--packet-trace`	Paquetes enviados y recibidos.

Ejemplo:

```
nmap -A -oA informe 192.168.1.1
```

Escaneo agresivo con salida en todos los formatos.

Reflexión final

Esta tabla no lo cubre todo, pero es una base útil para el día a día. Si necesitas más detalles, visita nmap.org.

> **Alternativa interesante:** Si buscas una herramienta similar pero enfocada en la velocidad, échale un vistazo a RustScan. Es extremadamente rápida y compatible con Nmap para escaneos más profundos.

Analizar tu servidor Linux con Lynis

Lynis es una herramienta muy útil para analizar sistemas Linux y Unix en busca de configuraciones inseguras, servicios innecesarios o fallos potenciales. Te da una radiografía bastante clara del estado de tu máquina, con recomendaciones para mejorarla.

Instalación y ejecución

Instalar Lynis:

```
git clone https://github.com/CISOfy/lynis.git
cd lynis
chmod +x lynis
```

Ejecutar auditoría completa:

```
sudo ./lynis audit system > lynis-report.txt
```

Esto generará un informe detallado que puedes consultar con tranquilidad.

Qué encontrarás en el informe

1. Arranque y servicios

Revisa qué servicios están activos y si hay algo inseguro desde el arranque.

```
Checking Secure Boot     [ DESHABILITADO ]
Checking running services (systemctl)   [ HECHO ]
Result: found 59 running services
```

> **Consejo:** Activa Secure Boot si puedes, y elimina servicios que no uses.

2. Seguridad del kernel

Opciones importantes del kernel y si requieren reinicio.

```
Check if reboot is needed     [ SÍ ]
```

> **Consejo:** Si te pide reiniciar, es por algo importante. Hazlo.

3. Usuarios y autenticación

Revisa cuentas, contraseñas y hashing.

```
Accounts without password     [ OK ]
Password hashing methods      [ SUGERENCIA ]
```

> **Consejo:** Usa SHA-512 y fuerza contraseñas fuertes.

4. Servicios en red

Qué interfaces están en modo promiscuo y qué puertos están abiertos.

```
Found promiscuous interface   [ NETW-3015 ]
Details: enp4s0
```

> **Consejo:** Si no necesitas que escuche todo, desactívalo.

5. Memoria y procesos

Procesos zombies, swap, uso de memoria.

```
Searching for dead/zombie processes   [ NO ENCONTRADO ]
Testing swap partitions               [ OK ]
```

> **Consejo:** Revisa qué corre en segundo plano.

6. Archivos y permisos

Archivos sensibles mal configurados.

```
Checking file permissions     [ SUGERENCIA ]
Deleted files in use          [ ARCHIVOS ENCONTRADOS ]
```

> **Consejo:** Vigila `/etc/passwd`, `/etc/shadow` y archivos eliminados aún abiertos.

7. Criptografía y red

Estado de SSL, cifrado, entropía del sistema.

```
Checking for expired SSL certificates [ NINGUNO ]
Kernel entropy is sufficient          [ SÍ ]
```

> **Consejo:** Usa cifrado fuerte y mantén certificados al día.

8. Conectividad

Configura bien tus DNS y evita puntos de fallo únicos.

```
Checking configured nameservers     [ PELIGRO ]
Minimal of 2 responsive nameservers [ PELIGRO ]
```

> **Consejo:** Añade varios DNS, no dependas de uno solo.

Conclusión

Lynis es una navaja suiza para auditar sistemas Linux. Es rápida, fiable y te da consejos directos. Lo ideal es usarla de forma periódica para detectar fallos antes de que sean un problema.

Referencias

- Sitio oficial de Lynis
- Repositorio en GitHub

Mejorando la seguridad de SSH con ssh-audit

Introducción

SSH es una de las puertas principales a tu servidor. Si la dejas mal cerrada, estás regalando las llaves de tu casa. Por suerte, con **ssh-audit** puedes revisar qué tan segura está y hacerle un buen repaso siguiendo prácticas modernas.

Características

Con `ssh-audit` puedes:

- Detectar configuraciones inseguras.
- Ver qué protocolos, cifrados, KEX y MACs están activos.
- Recibir recomendaciones concretas para mejorar tu setup SSH.

Instalación de ssh-audit

En Debian/Ubuntu:

```
sudo apt update && sudo apt install ssh-audit
```

En Fedora:

```
sudo dnf install ssh-audit
```

Instalación manual (si prefieres tirar de GitHub):

```
git clone https://github.com/jtesta/ssh-audit.git
cd ssh-audit
chmod +x ssh-audit.py
```

Cómo se usa

Para auditar otro servidor:

```
./ssh-audit.py <IP_o_dominio>
```

Para auditar tu propia máquina:

- Si lo instalaste desde GitHub:

```
python ssh-audit.py 127.0.0.1:22
```

- Si lo instalaste desde los repos del sistema:

```
ssh-audit 127.0.0.1:22
```

El resultado te dirá qué tan bien (o mal) tienes configurado el SSH, con colores y todo. Lo importante va en rojo. Ya sabes lo que eso significa.

Fortaleciendo SSH paso a paso

Edita la configuración principal:

```
sudo nano /etc/ssh/sshd_config
```

Ajustes clave que deberías tener:

- **Bloquear acceso directo como root:**

```
PermitRootLogin no
```

- **Solo protocolo 2 (el 1 ya está muerto):**

```
Protocol 2
```

- **Limitar a cifrados y algoritmos modernos:** (Quita las barras `/` y arrobas dobles `@@`, están ahí para evitar problemas de formato aquí)

```
Ciphers aes256-gcm@openssh.com,chacha20-poly1305@openssh.com
KexAlgorithms curve25519-sha256@libssh.org
MACs hmac-sha2-256,hmac-sha2-512
```

- **Autenticación por llaves públicas sí o sí:**

```
PubkeyAuthentication yes
```

Qué es la autenticación por llave pública (y por qué deberías usarla)

Adiós contraseñas, hola seguridad real. Funciona así:

- Tú tienes una **llave privada** que no sale de tu máquina.
- El servidor guarda la **llave pública** en `~/.ssh/authorized_keys`.

Cuando intentas conectarte, el servidor te deja pasar si las llaves encajan. Más seguro, más cómodo.

Cómo crear y usar llaves SSH

1. Genera tus llaves (usa tu email real):

```
ssh-keygen -t ed25519 -C "tu_email@example.com"
```

2. Sube la llave pública al servidor:

```
ssh-copy-id usuario@servidor
```

3. Prueba que ya no te pide contraseña:

```
ssh usuario@servidor
```

Reinicia el servicio SSH

Cada vez que cambies la config:

```
sudo systemctl restart ssh
```

Repite la auditoría de vez en cuando

No te olvides de pasar `ssh-audit` cada cierto tiempo. Los estándares cambian, y lo que hoy es seguro mañana puede estar obsoleto.

Para ideas más a fondo: Guías de hardening de SSH

Conclusión

Tener un servidor mal configurado es como dejar la puerta abierta y un cartel que diga “entra cuando quieras”. Dedica 10 minutos a esto, y tu SSH pasará de ser un blanco fácil a un hueso duro de roer.

Notas

Notas

Sistemas operativos y herramientas, orientados a ciberseguridad, anonimato y privacidad

Resumen de distribuciones y herramientas enfocadas en **pentesting**, **privacidad**, **OSINT**, **análisis forense**, **gestión de redes** y más. Algunas son distribuciones completas, otras son entornos virtuales o herramientas especializadas. El objetivo es tener una visión clara de qué usar según el caso.

1. Sistemas y herramientas para pentesting

- **Kali Linux**: Distro clásica para pruebas de penetración.
- **Parrot Security OS**: Ligera y versátil, orientada a pentesting y desarrollo seguro.
- **BlackArch Linux**: Miles de herramientas sobre Arch.
- **ArchStrike**: Similar a BlackArch, con enfoque ofensivo y defensivo.
- **Pentoo**: Basada en Gentoo, con buenas herramientas para auditoría.
- **Fedora Security Lab**: Variante oficial con herramientas de seguridad.
- **SamuraiWTF**: Framework para pruebas web.
- **Commando VM**: VM basada en Windows, orientada a entornos Windows.
- **BackBox**: Sencilla, usable y con herramientas de pentest.
- **Dracos Linux**: Ligera, enfocada en hacking.
- **Bugtraq**: Amplia gama de herramientas para auditoría y análisis.

2. Sistemas centrados en anonimato y privacidad

- **Tails**: Distro live con rutas por Tor.
- **Whonix**: Sistema virtualizado, anonimato extremo.
- **Qubes OS**: Seguridad mediante compartimentación.
- **Kodachi**: Con herramientas preconfiguradas para privacidad.
- **Subgraph OS**: Diseñada contra amenazas avanzadas.

---

3. Análisis forense y respuesta a incidentes

- **CAINE**: Para análisis y recuperación de datos.
- **Paladin Forensics**: Suite forense lista para usar.
- **REMnux**: Ideal para análisis de malware.
- **SIFT Workstation**: Sistema SANS para forense y respuesta.
- **Flare VM**: VM para análisis de malware y reversing.

---

4. Herramientas y distros para OSINT

- **Sleuth Kit Live**: Recopila y analiza evidencia digital.
- **Recon-ng**: Plataforma modular para OSINT.
- **OSINT Framework**: Mapa interactivo de herramientas.
- **Tsurugi Linux**: Completa, con enfoque en OSINT y análisis forense.

---

5. Plataformas para análisis y gestión de redes

- **Security Onion**: Detección de intrusos y respuesta a incidentes.

- **pfSense**: Firewall avanzado, muy estable.
- **OPNsense**: Derivado de pfSense, con mejoras visuales.
- **Smoothwall**: Solución comercial para filtrado y gestión.
- **VyOS**: Para routing avanzado y edge networking.
- **NethServer**: Para montar infraestructura segura fácilmente.
- **Untangle NG Firewall**: Freemium, con filtrado y análisis de tráfico.

---

6. Sistemas ligeros y minimalistas

- **Alpine Linux**: Muy ligera, ideal para contenedores o entornos embebidos.
- **Tiny Core Linux**: Extremadamente pequeña.
- **Slitaz**: Versátil para tareas específicas en hardware limitado.

---

7. Otros recursos útiles

- **Metasploitable**: VM intencionadamente vulnerable.
- **Hack The Box Academy**: Plataforma freemium para practicar pentest y hacking.

Nota final

Esta lista se irá ampliando y reorganizando según evolucione el ecosistema de ciberseguridad. Úsala como punto de partida para explorar herramientas según tus intereses o necesidades.