**INTRODUCCIÒN**

.

.

**ESTADO DEL ARTE**

Desde hace varios años ha ido en crecimiento el desarrollo de herramientas que nos ayuden en la gestión de la seguridad informática, existen productos robustos, maduros y muy respetables, no obstante, al irse presentando como toda una industria la ciberdelincuencia continúan en vanguardia de nuevos métodos, formas, tecnologías, que presenta todo un desafío a la seguridad en general.

Tanto individuos como empresas y universidades están dedicando mucho esfuerzo y trabajo en la investigación y desarrollo de la ciberseguridad, según indicó Canon Mexicana en el año 2020 afirmó que la ciberseguridad es el avance más importante dentro de las tecnologías de la información (IT) de los últimos años (Bibliografía # 1).

Como parte de este proyecto hemos investigado en Internet y encontrada información de artículos de reportajes de sitios de noticias de diferentes partes del mundo, sitios Web de proveedores de soluciones de seguridad informática, Wikipedia y otros.

Básicamente nuestra investigación nos ha llevado a concluir que se tienen todas las áreas de ciberseguridad cubiertas, sin embargo, hay algo que hemos encontrado y es que, para poder ampliar la capacidad de las herramientas, se depende del nivel de “velocidad” que el proyecto avance para contar con los últimos agregados o detecciones de nuevas amenazas, es decir hemos encontrado aquí un vacío a llenar.

Nuestra solución o herramienta en esencia se centraría en el “Information Gathering” enfocada en los administradores de TI y que sea fácilmente escalable.

Hablando en sí de las técnicas relacionadas del “Information Gathering”, podemos decir que la recolección de información es la capacidad de obtener la mayor cantidad de datos de un potencial blanco a auditar o atacar. La misma puede realizarse tanto de manera externa al blanco como interna, por tanto, podemos utilizar un número muy grande de herramientas disponibles para recolectar la mayor cantidad de información del equipo sin haber accedido a el, y por otro lado contamos con herramientas que nos brinda el propio sistema operativo.

Analizando las herramientas disponibles en Internet que permiten obtener y analizar la información de un servidor para detectar y corregir debilidades o vulnerabilidades, podemos diferenciar las mismas en cuatro categorías principales:

* Preventivas: Aquí encontramos herramientas que buscan fallas y una vez encontradas nos permiten conocer de manera detallada su solución (aplicar ‘X’ parche, remover ‘X’ usuario, etc)
* Reactivas: Estas herramientas se enfocan en tomar una acción al momento de detectarse un evento inusual que puede comprometer la seguridad del sistema. Podemos encontrar aquí los detectores de intrusos IDS (diferenciados como HIDS Host Intrusion Detection System, los cuales se enfocan en eventos a nivel del servidor como ser file System, integridad de archivos, programas, usuarios, procesos y NIDS Network Intrusion Detection System, enfocados exclusivamente en eventos de red). Toda la línea de soluciones “IDS” se enfocan en capacidad de detección y alerta, mientras que existe otra línea similar denominada “IPS” Intrusion Prevention System, la cual además de alertar, se encarga de tomar medidas correctivas (finalizar un proceso, bloquear una conexión, etc)
* Detección: En esta categoría podemos encontrar herramientas específicas de detección de una amenaza, compromiso o indicador que haga evidente un problema ya ocurrido en un servidor, pero no visible de manera normal.
* Recuperación: Herramientas dedicadas a recuperar frente a eventos ya ocurridos y que generaron daños en el equipo afectado, imposibilitando de acceder a un servicio, ejecutable, etc.

Si nos centramos un poco más en detalle, podemos encontrar para cada uno de los puntos, diversas áreas de cobertura. Si bien existen muchas opciones, detallamos aquí las más significativas, ya sea por popularidad o por haber tenido la oportunidad de utilizarlas. Entendemos que podemos separar las mismas de la siguiente manera:

**Análisis de vulnerabilidades**

Detallamos herramientas que permiten analizar equipos y detectar vulnerabilidades, ya sea por la versión del servicio expuesto o por la versión del software instalado. Entre las más destacadas mencionamos:

* **Nessus - https://tenable.com:** Considerada una de las pioneras en el análisis de vulnerabilidades, la misma se creó en el año 1998 por su fundador Renaud Deraison. Su objetivo fue que la comunidad tuviese una herramienta abierta para el escaneo de vulnerabilidades, aunque luego su licencia cambio. Al día de hoy la empresa propietaria es Tenable (https://tenable.com). Es una herramienta muy completa para el análisis de vulnerabilidades, la cual integra su propio lenguaje de scripting para escribir los diferentes scripts que componen las pruebas a realizar sobre un equipo auditado. Los scripts escritos para Nessus tienen la extensión NASL (Nessus Attack Scripting Language) y tienen un formato bastante simple (https://en.wikipedia.org/wiki/Nessus\_Attack\_Scripting\_Language )
* **Openvas - https://openvas.org:** Es una herramienta que nació como variante a Nessus en el año 2006, llamada inicialmente GNessus. Si bien al momento de realizar esta documentación, la herramienta está disponible en distribuciones de pentest y para descargar mediante paquetes y fuentes para compilar, nos parece que el proyecto luego de haber sido adquirido por la empresa GreenBone, está cambiando su rumbo, seguramente llegando a ser en cualquier momento un producto de licencia cerrada. Ejecuta los mismos scripts de Nessus (NASL) y cuenta con muchas funcionalidades a la hora de realizar un escaneo (https://es.wikipedia.org/wiki/OpenVAS). Actualmente cuenta con una versión gratuita y también se ofrece una versión paga derivado al cierre (Gratuita con restricciones - Paga)
* **Netsparker - https://www.netsparker.com:** Es una herramienta desarrollada inicialmente en el año 2006 por Ferruh Mavituna. La misma pone foco específicamente en el análisis de aplicaciones web independientemente de la tecnología, framework o lenguaje utilizado para su desarrollo, permitiendo escanear todos los puntos de la misma generando visualización muy completa del estado de seguridad de dicha aplicación a auditar.

**Auditoria - Hardening – Compliance**

* **https://cisofy.com/lynis/:** Esta es una herramienta gratuita muy completa, la cual permite realizar auditoria de seguridad, hardening y chequeos de compliance. Permite ejecutar de manera simple y rápida los chequeos necesitados. Una desventaja que tiene la herramienta es la no posibilidad de agregar chequeos de manera personalizada.

**Chequeo Integridad archivos/directorios e intrusión**

* **https://www.nongnu.org/tiger/:** Herramienta de código abierto con fuerte foco en un sistema HIDS (Host Intrusion Detection System). Cuenta con varios módulos de ejecución bastante interesantes, su última versión es del año 2018.
* **https://www.ossec.net/:** Se le puede considerar como la herramienta con mayor cantidad de funcionalidades y prestaciones para montar un servicio de detección y prevención de intrusos a nivel de host. Es un proyecto de código abierto desarrollado inicialmente por Daniel B. Cid. Actualmente se encuentra en desarrollo y goza de muy buena salud.
* **https://aide.github.io/:** Es un proyecto de detección de intrusos a nivel de chequeo de modificaciones de archivos y directorios. La versión actual es la 0.17.3 con fecha del 10/02/2021.

**Búsqueda de RootKits**

* **http://rkhunter.sourceforge.net/:** Proyecto iniciado en el año 2003 por M. Boelen, quien en el año 2006 entrego el mismo a un grupo de desarrolladores quienes se hicieron cargo desde la versión 1.2.9 (actualmente la última estable es la 1.4.6). La herramienta se encarga de analizar el equipo sobre el cual se ejecuta a efectos de buscar señales de compromiso, como ser binarios alterados, procesos en memoria no reflejados en lista, ejecutables conocidos, etc.
* http://www.chkrootkit.org/: Se lo puede considerar como una versión mejorada de rkhunter, desarrollado por Nelson Murilo y Klaus Steding Jessen. La versión actual es la 0.55 con fecha de lanzamiento 11/06/2021 y su foco es la búsqueda de alteraciones a nivel del servidor (de todo nivel, interfaces de red, procesos, binarios, etc) a efectos de diagnosticar un compromiso.

**Administración de vulnerabilidades**

**https://www.open-scap.org/tools/openscap-base/:**  SCAP es un método de utilización de estándares específicos que permite automatizar la administración de vulnerabilidades (https://en.wikipedia.org/wiki/Security\_Content\_Automation\_Protocol). La herramienta permite asistir tanto a administradores como auditores en la evaluación, medición y aplicación de las líneas de base de seguridad. Para tener un detalle exacto de la definición de SCAP, se sugiere el siguiente enlace: https://public.cyber.mil/stigs/scap/

**MOTIVACIÓN**

Considerando la multitud , prestancia y relevancia que ofrecen los servidores Linux a nivel global, y lo vital de mantenerlos seguros, nos hemos sentido motivados a contribuir con un granito de arena en poder suministrar una herramienta que ayude a los administradores de arquitecturas en Linux a poder evaluar los riesgos de seguridad de sus implementaciones, y a poder “auto corregirse” y de minimizar de manera preventiva alguna configuración, o niveles de permisos que pudieran constituirse en posibles puertas de acceso de ciber delincuentes o que permitan fuga de información, etc.

Nos llamó de sobremanera la atención lo módulos: 4: Seguridad en Arquitecturas Linux y 5: Seguridad en Redes corporativas, dado la practicidad de su contenido y el enfoque del tutor, todo ello nos llevó a querer basar nuestra tesis en estos atractivos módulos del master.

Realizando un análisis de mercado encontramos la siguiente información referente a Linux, lo cual aportó también a dirigirnos por una solución apuntando a este segmento:

* El 96,3% del millón de servidores más importantes del mundo se ejecutan en Linux. (Bibliografía # 2)
* El 90% de la carga de trabajo de la nube pública se ejecuta en Linux. (Bibliografía # 1)
* En 2021, el 100% de las 500 supercomputadoras más importantes del mundo funcionan con Linux. (Bibliografía # 2)
* El 36,7% de los sitios web con sistemas operativos conocidos utilizan Linux en 2021. (Bibliografía # 2)
* El 83,1% de los desarrolladores dice que Linux es la plataforma en la que prefieren trabajar. (Bibliografía # 2)

Es indiscutible que cada día se hace más relevante la seguridad informática y las organizaciones deben estar muy alertas y tomar medidas preventivas que les ayuden a minimizarlas, a continuación, mencionamos ataques o información pertinente del 2021 que han alimentado nuestra motivación en este proyecto:

* Ecuador: Ataque a la corporación nacional de Telecomunicaciones (CNT) – (Bibliografía 3,4,5)

Fue declarada en emergencia institucional por los ataques informáticos que se inicio el 14 de Julio de 2021, el ransomware EXX. El daño provocó que los empleados de la empresa pública apagaran sus computadoras. Este tipo de virus puede atacar diversas instancias: el sitio, la red o las bases de datos, en este el ataque fue directamente a los sistemas informáticos internos de la empresa como facturación, activaciones y recargas.

La ministra del Ecuador no se refirió a quienes estarían detrás del ataque a CNT. Sin embargo, dijo que no es un tema económico el que motivó a los “criminales” a hacer esto. Dijo también que “es un hackeo costoso, de altísimo nivel” y que se investigará a los autores nacionales y extranjeros que estén involucrados.

* Uruguay: Ciberataques contra el Estado: un nuevo escenario para la defensa: (Bibliografía 6)

En Uruguay dos “inéditos” ataques vulneraron a reparticiones estatales en los últimos meses. La Dirección de Identificación Civil fue víctima en diciembre y la Armada Nacional sufrió una extorsión virtual en enero.

El 15 de enero, la Armada Nacional informó en su cuenta de Twitter que había sido víctima de un ciberataque. Habían identificado un mail externo que contenía datos sobre 50 cuentas de funcionarios de la Armada. Un hackeo “inédito” en la institución según expresó su vocero, Pablo González Cambón.

El lunes 1ero. de febrero, el ministro de Defensa Nacional, Javier García, compareció ante los legisladores de la Comisión de Defensa del Senado, donde brindó algunos detalles sobre el ciberataque que sufrió la Armada Nacional a manos de un hacker “presuntamente de origen ruso”. García explicó que se trató de una maniobra de tipo “phishing” y que los hackers se hicieron de la información a través de un correo electrónico que enviaron a un funcionario de la Armada.

* EEUU: "Colosal" ciberataque golpea a cientos de empresas en EE. UU (Bibliografía 7)

La firma Kaseya reconoció que una de sus aplicaciones que administra servidores corporativos, computadoras de escritorio y dispositivos en red pudo haber estado comprometida en el ataque.

La compañía dijo que estaba alertando a sus clientes que usan su herramienta VSA a que apaguen inmediatamente sus servidores.

En su comunicado, Kaseya aseguró que un "pequeño número" de empresas habían sido afectadas.

La empresa de ciberseguridad Huntress Labs afirmó que el objetivo del ataque fue la compañía de tecnología informática Kaseya, basada en Florida, y luego se extendió por las redes corporativas que usan su software, asegurando afirma que la cifra es más que 200 empresas clientes de Kaseya.

* Crimen como Servicio 2021: secuestro de datos bajo demanda (Bibliografía 8)

Entre los principales servicios que pueden obtenerse como CaaS, destacan:

Kits/plataformas de phishing: disponibles en la Dark Web por tan solo 2 o 10 dólares, estos kits y plataformas se pueden personalizar con pocos conocimientos y tienen varios niveles de automatización, lo que los hace muy atractivos para los delincuentes.

Kits de exploits: Incluyen el desarrollo de código de explotación y herramientas para explotar vulnerabilidades conocidas. Uno de los más populares, RIG, cuesta 150 dólares/semana y puede propagar ransomware, troyanos y otras formas de malware.

Servicios DDoS: Los servicios DDoS son baratos y accesibles, y muchos proveedores ofrecen planes de suscripción en la Dark Web. Otros realizan ataques DDoS a servidores o sitios web que utilizan protección, y algunos incluso ofrecen ataques a recursos guberna-mentales específicos.

Ransomware como servicio: Estos servicios proporcionan la profundidad técnica y las habilidades requeridas, además de toda la información necesaria para llevar a cabo un ataque. En algunos casos, ofrecen un panel de control e informes sobre su estado.

Investigación como servicio: Implica la recopilación legal o ilegal de información sobre las víctimas objetivo, así como la reventa de los datos personales robados, como las credenciales comprometidas. Puede incluir la venta de información sobre posibles exploits dentro de software o sistemas.

**HIPÓTESIS**

Nuestro proyecto toma en base a los datos levantados y análisis del alcance del proyecto y soluciones del mercado que podremos aportar de manera significativa a los administradores de TI en arquitecturas en Linux con una herramienta sencilla y eficaz que les permitirá obtener sugerencias realistas de mejoras de seguridad en sus servidores Linux.

La aspiración es que la herramienta analice logs, obtenga servicios levantados, con que usuarios son ejecutados, niveles de acceso de dichos usuarios, realización de varias pruebas con los usuarios encontrados para determinar posibles brechas de seguridad o expuestas a ataques y sugerencias.

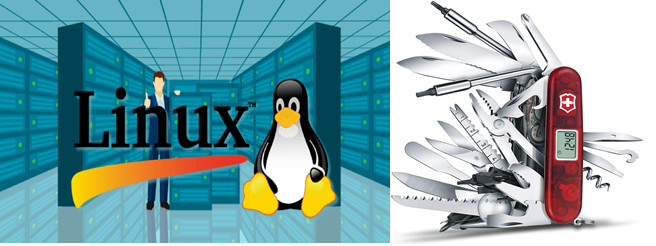
Se parte con la premisa de que a mayor nivel de acceso de usuario utilizado para el Login de le herramienta la misma podrá analizar a mayor profundidad los servidores especificados. Una ejecución recursiva tratando de “emular” a los usuarios encontrados pudiera bien darnos una foto de la capacidad de acceso, acción y maniobrabilidad que los usuarios tendrían de los recursos y servicios del servidor.

Conceptualmente hablando nuestra herramienta debe ser capaz de mediante los usuarios encontrados saber que podría hacer un atacante si pudiera ejecutar software con nivel de acceso similar y en base a ello poder diagnosticar o predecir qué tipo de ataque pudiera realizar y dar soluciones o guía a los administradores para que puedan evitarlo o mitigarlo.

Pretendemos poder almacenar la información recopilada (los comandos ejecutados para ello) en una base de datos y poder o bien graficarla o presentarla en un reporteador propio o en su defecto crear un plugin o interfaz con alguna herramienta de reportería de terceros.

La herramienta pretende permitir sea a los usuarios/administradores, o a nosotros como desarrolladores del producto añadir script/comandos o funciones adicionales para ir robusteciendo a la herramienta en el tiempo, permitiendo su fácil crecimiento, alentando a crear una comunidad que pueda alimentarla y darle la flexibilidad que nos gustan tener sobre las soluciones de TI.

Considerando la relevancia de Linux como servidor en servicios importantes para las organizaciones, y el vacío encontrado en alguna herramienta fácilmente escalable para que los administradores puedan evaluar sus servidores estamos convencidos que nuestro proyecto será de gran utilidad, aspirando sea convierta en el “victorinox” del information Gathering:



**TESIS**

Nuestra tesis empezó con la revisión de la Presentación PFMs y documentación sobre los diversos proyectos que podríamos realizar como parte del fin del MASTER.

Alejandro Martínez tomo la iniciativa de contactar a su compañero del MASTER Angel Mera para pode formar entre los dos un equipo que trabaje junto en la tesis, en una primera reunión definieron que les vendría muy bien la creación de alguna herramienta considerando los conocimientos de programación de ambos, dada también la experiencia y trabajo a diario de Alejandro en Linux y de administración de servidores Linux que posee Ángel, se inclinaron en basar su proyecto en los Módulo 4 y 5: Seguridad en Arquitecturas Linux y Seguridad en Redes corporativas.

**La repartición del trabajo se definió de manera unánime de la siguiente forma:**

**Alejandro Martínez:**

1. Estado del arte: estado actual en este rubro
2. Técnicas relacionadas con "Information Gathering"
3. Artículo en formato entrada de blog

**Ángel Mera:**

1. Motivación
2. Hipótesis
3. Presentación en PowerPoint

**En conjunto:**

Tesis.

Desarrollo de la herramienta y reporte.

Demostración en video del trabajo si procede.

Dado que este proyecto está pensado para administradores y es de suponer que el uso de scripts es algo de uso habitual y común en las tareas diarias, que mejor opción que permitirle al propio administrador escribir sus scripts y que estos sea el punto de entrada para la herramienta, permitiendo:

1. Generar scripts propios que se encarguen de realizar tareas específicas a la necesidad puntual

2. No depender de la velocidad de lanzamiento de nuevas actualizaciones

3. Simplificar el desarrollo en un lenguaje de scripting muy simple como es bash

Hay que tener presente que cuando se ataca y explota un servicio determinado, este corre bajo el ID de un usuario puntual, con lo cual, cuantas más herramientas tengamos disponibles para utilizar, mayor recolección de información vamos a poder lograr.

Linux nos ofrece una cantidad de herramientas y utilidades ya instaladas por defecto, las cuales perfectamente pueden servirnos (y van a servirnos) para este proyecto.

Para poder tener claro la información que necesitamos recolectar, primero tenemos que tener claro que es lo que vamos a revisar, con lo cual vamos primero a separar en diferentes capas de acceso.

Tomando como base principal los datos a proteger (la cual sería la capa “interna”) a partir de ahí nos vamos alejando hasta la capa más “externa” que es la red.

Hecha la aclaración tenemos el siguiente diagrama:



Descripción de las capas y herramientas disponibles en cada una de ella

**Red:** Puede variar de acuerdo a la red donde se encuentre el equipo. Lo importante es que aquí tenemos varias opciones a realizar a efectos de mejorar la misma.

Algunas herramientas disponibles son:

* netstat / ss: Ver conexiones activas, servicios en escucha, servicios por protocolo, estado de conexiones, procesos asociados, contextos, enrutamiento, etc
* route: Tabla de ruteo
* ip / ifconfig: información de las interfaces de red, estado, errores, dirección MAC, túneles, ruteo, etc
* arp: Tabla de direcciones MAC
* ethtool: Estado de las interfaces de red, driver asociado, estadísticas, configuraciones

**Usuarios:** Esta capa refiere a los permisos de acceso de los usuarios en el sistema e incluye autentificación, autorización y acceso.

Hay muchos controles aquí que implican verificaciones a nivel de permisos. Se mencionan algunos ejemplos:

* Usuarios con id 0
* Terminales habilitadas para realizar login
* Cuentas con o sin shell
* Grupos a los cuales pertenece un usuario
* Restricciones mediante PAM (fortalecimiento de contraseñas, forzar cambios, historial, etc)

**Sistema Operativo:** En este punto, hacemos referencia a todo lo que implica el sistema operativo auditado o comprometido. Esto incluye procesos en ejecución, servicios por defecto habilitados, aplicaciones y su configuración, manejo de dispositivos, etc.

Ejemplos de control:

* Logs de auditoría registrados y guardados en sistema centralizado
* Revisión de violaciones de acceso
* Permisos de ejecución de herramientas sensibles (o peligrosas)
* Control de accesos a comandos como root (su)
* Dispositivos y herramientas del sistema protegidos contra usuarios no autorizados (sniffers, analizadores de protocolo, herramientas de red, etc)

**Sistema de Archivos:** Es importante restringir los permisos de acceso de los usuarios a determinados lugares del sistema de archivos que no deben acceder. Ejemplos:

* Registros de sucesos.
* Archivos y directorios sin propietario.
* Permisos de acceso a archivos de configuración.
* Minimizar los directorios del sistema operativo con permisos totales.

**Datos:** Esta es la información que estamos protegiendo. Debemos tener claro que protegemos y como lo estamos haciendo, Ejemplos:

* Si es una base de datos, permisos de archivos, usuarios, contraseñas, accesos anónimos, bases de test habilitadas.
* Archivos de texto, permisos mencionados anteriormente

Todos los puntos mencionados anteriormente parten de la base de tener acceso físico al servidor a auditar, no obstante, siempre hay un punto previo que implica el análisis y recolección de toda la informacion posible de manera pasiva, sin haber accedido aun al servidor.

Para esto, podemos dividir en diferentes “capas” de análisis como ser:

* Obtención de informacion de datos públicos.
* Análisis de servicios expuestos (versiones detectadas y problemas conocidos)
* Detección de los servicios (escaneo de puertos, consulta en bases públicas)
* Análisis de vulnerabilidades (Herramientas)
* Análisis de topología (Detección de conectividad, Gateway, red, etc)

**Especificaciones técnicas de la herramienta:**

Leguajes de programación utilizados: **PYTHON 3.8 Y SCRIPT BAJO SHELL DEL LINUX.**

Se escogió PYTHON como lenguaje de programación dada su sencillez, potencia, limpieza, legibilidad, a más de su fácil integración o capacidad de ejecutar script que se ejecutan bajo el shell de Linux.

Está desarrollado bajo una licencia de código abierto, por lo que es de libre uso y distribución, incluso para uso comercial. PYTHON cuenta con una muy buena comunidad, que cuida el lenguaje y casi todas las actualizaciones se hacen de manera democrática

Base de datos: **SQLITE**.

La selección de esta base de datos se debe a lo siguiente:

Está basado en archivos y fácil de configurar y utilizar. Adecuado para el desarrollo básico y pruebas.

Utiliza sintaxis SQL estándar con pequeñas modificaciones.

Es de código abierto, mucha de su información se puede encontrar disponibles para toda la comunidad.

Requisitos de Software:

* Linux cualquier distribución.

Requisitos de Hardware:

* Pentium 4 en adelante.
* Arquitectura de 32 o 64 bits.
* 512 de RAM en adelante
* 3 GB de espacio en disco duro.

**Diagrama de la herramienta:**

Programa principal:

Carpeta con los Script a ejecutar.

Ejecución del análisis

Grabar LOG del análisis

Base de Datos de tareas y script.

Generación de reportes

En formato HTML

En formato CSV

**Estructura del Proyecto:**

Carpeta principal TFM1.

Subcarpetas:

[**database**](https://github.com/alejandroguardian/TFM11/tree/6f5a6485932bfb26ec3a94f66d02174a8c60ffe8/database)**:** Posee las bases de datos de SQLITE llamada m11tfm.db

[**docs**](https://github.com/alejandroguardian/TFM11/tree/6f5a6485932bfb26ec3a94f66d02174a8c60ffe8/docs)**:** Documentación del proyecto.

**f**[**uentes**](https://github.com/alejandroguardian/TFM11/tree/6f5a6485932bfb26ec3a94f66d02174a8c60ffe8/fuentes)**:** Código fuente de los archivos PYTHON

Posee las siguientes subcarpetas:

**\_\_pycache\_\_:** Carpeta de uso de caché.

**reportes:** Posee los archivos CSV y HTML que se generan como reportes al momento de ejecutar el análisis.

**Scripts:** Contiene los scripts .sh que se vinculan a cada tarea que forma parte del análisis (cuando se cree un nuevo análisis, se debe a más de crearlo en la base de datos como una nueva tarea, copiar aquí el respectivo archivo .sh).

**Contenido de la base de datos de SQLITE llamada m11tfm.db**

Tablas:

**Tasks**.- Contiene las diferentes tareas que se realizarán como parte del análisis, su estructura es:

**id** Integer PRIMARY KEY AUTOINCREMENT.- Identificador de la tarea.

**name** Text.- Descripción de la tarea

**cmd** Text.- Nombre del Script a ejecutar.

**result** Text.- Resultado esperado.

**description** Text .- Detalle de la tarea.

**Output**. - Posee el historial de ejecución de la herramienta, grabando el resultado de cada análisis ejecutado, su estructura es:

**id** Integer PRIMARY KEY.- Identificador del registro.

**ItemId** Integer.- Identificador de la tarea.

**ItemResult** Integer.-1=Indica posible problema de seguridad o resultado verdadero, 0=Retorna resultado idóneo o respuesta falsa.

**ItemResultDetail** Text.-Detalle del resultado.

**Job** Numeric.- Identificador de la transacción.

**Date** Datetime.- Fecha y hora.

**Programas creados:**

Dentro de la carpeta fuentes:

* analisis.py.- Ejecuta el análisis , invocando a los diferentes scripts .SH parametrizados, también crea los archivos .SH y .HTML que muestran los resultados del análisis.
* db.py.- Administra las conexiones con la base de datos, permite ejecutar las sentencias SQL.
* reporte.py.- Programa que permite la visualización de los archivos .CSV.
* reportecsv.py.- Programa que genera los archivos CSV
* reportehtml.py.- Programa que genera los archivos HTML
* tfm11.py.- Main o programa principal, es el que se debe ejecutar para arrancar el análisis. Cabe aclarar que hay una opción no implementada que es la de agendar. La misma se desarrollara a posteriori y la idea es implementar un simple script que mediante un cron corra cada 1 minuto y verifique si se da el “match” de día – hora – minuto y si es así, ejecuta el script tfm11.py con la opción “e”

**Código Fuente de los programas PYTHON**

**tfm11.py**

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

# Imports del sistema

import os

import sys

import getopt

# Archivos propios

import db as db

import analisis as a

import reporte as r

# Mensaje de uso

def uso():

msg = '''

####################################################################

# TFM - Trabajo Final de Maestria - Alejandro Martinez, Angel Mera #

####################################################################

Uso: ./tfm.py opcion

Opciones

a - Agendar analisis

e - Ejecutar analisis

r - Acceso Reportes

'''

print(msg)

# Main

def main():

if ( len(sys.argv) == 1 ):

uso()

else:

param = sys.argv[1]

if param == "a":

#agendarAnalisis()

print("Agenda - No disponible")

elif param == "e":

a.ejecutarAnalisis()

elif param == "r":

r.ejecutarReporte()

else:

print ("Parametro Incorrecto")

sys.exit(2)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**analisis.py**

*#!/usr/bin/python*

*# -\*- coding: utf-8 -\*-*

*# Este script se encarga de ejecutar un nuevo analisis*

*# Imports*

*import db*

*import reportehtml as outhtml*

*import os*

*import reportecsv as outcsv*

*from datetime import datetime*

*from random import \**

*# Rutina de analisis*

*def ejecutarAnalisis():*

*# Identificador unico del reporte*

*repid = str(randint(1,1000000))*

*# Reporte CSV*

*dt = datetime.now()*

*filename = 'reportes/' + dt.strftime('%Y%m%d%H%M') + '.csv'*

*# filename = 'reportes/' + repid + '.csv'*

*ocsv = outcsv.OutputCsv()*

*ocsv.abroLog(filename)*

*# BD - Lectura de tareas a ejecutar*

*db.opendb()*

*data = db.conn.execute('select name,cmd,result,id from Tasks')*

*# Parseo tareas*

*ocsv.escriboCabezal('Reporte,Control,Resultado Esperado,Resultado Obtenido,Status\n')*

*# Realizo looop en la tabla "Tasks"*

*cont=0*

*for row in data:*

*name = row[0]*

*cmd = 'scripts/' + row[1]*

*result = row[2]*

*itemid = str(row[3])*

*# Inicio*

*print("["+str(cont)+"] Ejecutando: "+name, end='')*

*cont += 1*

*# Ejecuto Script*

*stream = os.popen(cmd)*

*cmdout = ''.join(stream.read().split())*

*# Verifico resultado obtenido (cmdout) contra salida esperada (result)*

*if cmdout == result:*

*print(" - OK") # Todo OK - Registro en archivo*

*# Registro*

*dt = datetime.now()*

*db.conn.execute("INSERT INTO Output (Job,Date,ItemId,ItemResult,ItemResultDetail) VALUES ("+repid+ ",'"+str(dt.strftime('%Y%m%d%H%M'))+"','"+itemid+"',0,'OK')");*

*else:*

*print(" - ERR") # Error - Vuelvo a correr el script pasandole "D" por linea de comando ("D"etalle)*

*stream = os.popen(cmd + " D")*

*cmdout = ' '.join(stream.read().split())*

*if not cmdout:*

*cmdoutshow = "Not found"*

*dt = datetime.now()*

*db.conn.execute("INSERT INTO Output (Job,Date,ItemId,ItemResult,ItemResultDetail) VALUES ("+repid+ ",'"+str(dt.strftime('%Y%m%d%H%M'))+"',"+itemid+",1,'"+cmdout+"')");*

*# Grabo BD*

*db.conn.commit()*

*# Variable para el archivo de Log*

*grabo = repid + ',' + str(name) + ',' + str(result.encode('utf-8')) + ',' + str(cmdout.encode('utf-8'))*

*# Verifico la salida*

*if cmdout == result:*

*grabo = grabo + ',OK'*

*else:*

*grabo = grabo + ',ERR'*

*# Grabo Log*

*ocsv.escriboLog(grabo + '\n')*

*db.closedb()*

*ocsv.cierroLog()*

*# Generacion de html*

*ohtml = outhtml.OutputHtml()*

*ohtml.armoHtml(repid)*

**db.py**

*#!/usr/bin/python*

*# -\*- coding: utf-8 -\*-*

*# Modulo de trabajo con Base de Datos*

*import sqlite3*

*# Ejemplos*

*#conn.execute("INSERT INTO Tasks (name,cmd,result,description) VALUES ('Check root user', 'cat /etc/passwd | grep :0: | wc -l', '0', 'Find number of users with id 0' )");*

*#conn.commit()*

*#print ("Records inserted successfully")*

*# Apertura DB*

*def opendb():*

*# Variables*

*global conn*

*# Conexion a la base*

*database = '/home/alejandro/Documentos/maestria/M11\_TFM/trabajo/repo/database/m11tfm.db'*

*# Abro DB*

*try:*

*conn = sqlite3.connect(database)*

*print("Conectado a BD")*

*#except sqlite3.Error as error:*

*except Exception as e:*

*print("Error de conexion a BD", e)*

*# Cierre DB*

*def closedb():*

*print("Cerrando DB")*

*conn.close()*

**reporte.py**

*#!/usr/bin/python*

*# -\*- coding: utf-8 -\*-*

*# Este script se encarga de ejecutar reportes*

*# python3 -m pip install simple-term-menu*

*# Imports*

*from simple\_term\_menu import TerminalMenu*

*import os*

*import pandas*

*# Clase principal*

*def ejecutarReporte():*

*# Borro pantalla*

*if os.name == 'posix':*

*\_ = os.system('clear')*

*else:*

*# Windows*

*\_ = os.system('cls')*

*msg = '''*

*####################################################################*

*# TFM - Trabajo Final de Maestria - Alejandro Martinez, Angel Mera #*

*####################################################################*

*'''*

*print(msg)*

*print("Seleccione un analisis")*

*ruta = "./reportes/"*

*dirs = os.listdir(ruta)*

*files = [x for x in dirs if os.path.isfile(os.path.join(ruta, x))]*

*terminal\_menu = TerminalMenu(files)*

*menu\_entry\_index = terminal\_menu.show()*

*reporteCsv = pandas.read\_csv("reportes/"+str(files[menu\_entry\_index]), usecols =["Control","Resultado Esperado","Resultado Obtenido","Status"])*

*print(reporteCsv)*

**reportecsv.py**

*#!/usr/bin/python*

*# -\*- coding: utf-8 -\*-*

*# Este script se encarga de grabar el analisis como Csv*

*# Clase de escritura a archivo*

*class OutputCsv:*

*# Abro Archivo*

*def abroLog(self, filename):*

*self.f = open(filename, 'w')*

*# Escribo Cabezal*

*def escriboCabezal(self, cabezal):*

*self.f.write(cabezal)*

*# Escribo log*

*def escriboLog(self, msg):*

*self.f.write(msg)*

*# Cierro Log*

*def cierroLog(self):*

*self.f.close()*

**reportehtml.py**

*#!/usr/bin/python*

*# -\*- coding: utf-8 -\*-*

*# Este script se encarga de grabar el analisis como html*

*from datetime import datetime*

*import db*

*import reportecsv as outcsv*

*# import webbrowser*

*# Clase de escritura a archivo*

*class OutputHtml:*

*def armoHtml(self, repid):*

*dt = datetime.now()*

*filename = 'reportes/html/' + dt.strftime('%Y%m%d')+'-'+dt.strftime('%H%M') + '-'+ repid + '.html'*

*ocsv = outcsv.OutputCsv()*

*ocsv.abroLog(filename)*

*db.opendb()*

*mid=str(repid)*

*mid=mid.strip()*

*cadena="select tas.name,det.ItemResultDetail result,det.ItemResult output,det.date date\_time,det.ItemId from Output det join Tasks tas on tas.id=det.ItemId where det.Job="+mid*

*data2 = db.conn.execute(cadena)*

*vmensaje="""<br><div style='border: 0pt solid #bfb;'>*

*<table border='1' align='center'>*

*<tr>*

*<td><font color='Zian'>INFORME EJECUTIVO</font></td>*

*</tr>*

*</table>*

*</div>"""*

*vmensaje=vmensaje+' '+'<BR><TABLE BORDER=2 align="center" bordercolor="blue" >'*

*vmensaje=vmensaje +'<TR>'+'<TD align="center">'+'<font color="Maroon">'+'#'+'</font>'+'</TD>'+'<TD align="center">'+'<font color="Maroon">'+'Descripción del Proceso'+'</font>'+'</TD>'+'<TD align="center">'+'<font color="Maroon">'+'Resultado'+'</font>'+'</TD>'+'<TD align="center">'+'<font color="Maroon">'+'Salida'+'</font>'+'</TD>'+'<TD align="center">'+'<font color="Maroon">'+'Fecha y Hora'+'</font>'+'</TD>'+'</TR>'*

*for row2 in data2:*

*vdatetime=str(row2[3])[0:4]+'/'+str(row2[3])[4:6]+'/'+str(row2[3])[6:8]+' '+str(row2[3])[8:10]+':'+str(row2[3])[10:12]*

*vmensaje=vmensaje +'<TR>'+'<TD align="center">'+'<font color="Maroon">'+str(row2[4]).strip()+'</font>'+'</TD>'+'<TD align="center">'+'<font color="Maroon">'+row2[0]+'</font>'+'</TD>'+'<TD align="center">'+'<font color="Maroon">'+row2[1].strip()+'</font>'+'</TD>'+'<TD align="center">'+'<font color="Maroon">'+str(row2[2])+'</font>'+'<TD align="center">'+'<font color="Maroon">'+vdatetime+'</font>'+'</TD>'+'</TR>'*

*vmensaje=vmensaje+'</TABLE>'*

*ocsv.escriboLog(vmensaje + '\n')*

*ocsv.cierroLog()*

*db.closedb()*

*#Abriendo archivo HTML con el Navegador*

*# webbrowser.open\_new\_tab(filename)*

**Controles efectuados por los scripts**

**Red**

|  |  |
| --- | --- |
| Chequear servicios innecesarios | Servicios no necesarios no deben estar ejecutando |
| Asegurar el dueño de los servicios | El dueño de los servicios debe ser root. Verificar que solamente root pueda escribir en ellos |
| Asegurar servidor de impresión | /etc/cups/cupsd.conf  Permissions = 600  Owner = lp  Group = sys |
| Archivo rhosts | Asegurarse que no existen .rhosts en el sistema |
| No se deben definir equipos de confianza | No debe existir el archivo /etc/hosts.equiv |
| Interfaces ppp no deben existir | Pueden bypasear el firewall |
| Parámetros de red correctamente configurados | *Revisar en /etc/sysctl.conf*  *o net.ipv4.tcp\_syncookies = 1*  *o net.ipv4.conf.all.rp\_filter = 1*  *o net.ipv4.conf.all.accept\_source\_route = 0*  *o net.ipv4.conf.all.accept\_redirects = 0*  *o net.ipv4.conf.all.secure\_redirects = 0*  *o net.ipv4.conf.default.rp\_filter = 1*  *o net.ipv4.conf.default.accept\_source\_route = 0*  *o net.ipv4.conf.default.accept\_redirects = 0*  *o net.ipv4.conf.default.secure\_redirects = 0*  *o net.ipv4.icmp\_echo\_ignore\_broadcasts = 1*  *o net.ipv4.ip\_forward = 0*  *o net.ipv4.conf.all.send\_redirects = 0*  *o net.ipv4.conf.default.send\_redirects = 0*  *o net.ipv4.icmp\_ignore\_bogus\_error\_responses = 1* |
| Deshabilitar IPv6 | Verificar que IPv6 no este activo |
|  |  |

**Usuarios**

|  |  |
| --- | --- |
| No deben existir cuentas con ID 0, solo root. | Verificar /etc/passwd |
| Restringir logins de root solamente a consola | Verificar /etc/securetty (console) y /etc/ssh/sshd\_config (PermitRootLogin no) |
| Cuentas del sistema no deben poder realizar login interactivo | Cuentas con UID < 500 verificar “shell=/sbin/nologin”  /etc/shadow agregar “!” al campo “password” |
| Las cuentas no deben tener UID duplicados | Verificar el archivo /etc/passwd que no contengan UID duplicados |
| .- No se deben asignar contraseñas a los ID de grupo.  .- Los nombres e ID de los grupos deben ser únicos Implementación  .- Los grupos deben contener usuarios existentes válidos | Confirmar que el archivo /etc/group esta formateado correctamente mediante el archivo /usr/sbin/grpck. El archivo debe contener 4 campos de datos.  Asegurarse que los nombres de grupo y GID sean únicos en el sistema  Asegurarse que los grupos solo contentan usuarios existentes válidos |
| Bloquear cuentas luego de 3 intentos de login no exitosos | Verificar PAM |
| Todas las cuentas de usuario deben tener un shell y un home único | Todos los usuarios deben teber un shell válido (/etc/passwd)  Verificar los permisos de los directorios home no deben ser mayores a 755 (rwxr-xr-x) |
| El archivo .forward no debe existir | Asegurarse que el archivo .forward no existe en los directorios de los usuarios |
| Encriptación de las contraseñas | /etc/passwd , verificar que cada entrada tenga una “x” para indicar que están activas las shadow passwords |
| Cuentas con passwords vacíos | Verificar /etc/shadow que no exsitan passwords vacíos |
| Comando xhost | No debe existir el comando “xhost +” en los archivos de inicialización de los usuarios |
| Archivos “.exrc” | Asegurarse los archivos “.exrc” no contengan secuencias “!” |
| Grupo “disk” | Verificar que ningún usuario pertenezca al grupo “disk” |

**Procesos**

|  |  |
| --- | --- |
| Path | Verificar que el directorio actual esta removido del PATH |
| Tiempo de inactividad | Verificar en /etc/profile que exista una sentencia “tmout” |
| Mostrar ultimo login exitoso | Verificar existencia del archivo “/var/log/lastlog” |
| Sistema gráfico | Si no es necesario, verificar que no esté ejecutando el modo grádico (runlevel 5) |
| Montajes nfs | “/etc/fstab” verificar que exista la opción “nosuid” |
| Montajes removibles | “/etc/fstab” verificar las opciones “nosuid” y “nodev” para dispositivos removibles |
| Homes | “/etc/fstab” verificar las opciones “nosuid” y “nodev” para los homes |
| Deshabilitar dispositivos de almacenamiento USB | Verificar en “/etc/modprobe.conf” o lo que esta debajo exista “blacklist usb-storage” |
| Servidor NFS | Si el sistema ofrece servicio nfs, verificar lo siguiente:  “no\_root\_squash” no este presente  “no\_acl” no este presente  “anonuid” sin UID 0. Verificar que el UID sea válido  Verificar que el “/” filesystem no sea exportado  Verificar que existan límites de exportación |
| Logs de autentificación | El archivo “/var/log/secure” o “/var/log/auth.log” debe existir  Dueño: root  Grupo: root  Permisos: 600 |
| Deshabilitar recepción de logs (syslog) si no es necesario | Verificar que no esté en escucha el servicio syslog |
| Deshabilitar montaje de filesystems de usuario | “/etc/security/console.perms” Comentar entradas referidas a floppy y unidades de CD |
| Proteger GRUB con contraseña | “/boot/grub/grub.cfg” Verificar que contenga una sección “password” |
| Auditar inicio y detención de servicios | Asegurarse los inicios y detenciones de servicios queden registrados |
| Auditar comando “su” para obtener root | Verificar que “autpriv.\* /var/log/secure” este activo |
| Verificación directorio “lost+found” | Debe estar vacío |
| Formato del archivo “/etc/passwd” | Debe estar correctamente formateado. Debe contener 7 campos de datos (columnas) |
| Entradas “legacy” no deben existir (+) | Confirmar que el signo “+” no existe en  “/etc/passwd”  “/etc/shadow”  “/etc/group”  “/etc/gshadow” |
| Login shell, PATH completo | Asegurarse el shell de login en “/etc/passwd” sea especificado con PATH absoluto |
| Archivo de profile debe existir | Verificar que el archivo “/etc/skel/.profile”  existe |
|  |  |

**File System**

|  |  |
| --- | --- |
| Dueños de archivos y directorios | Verificar que no existan archivos y directorios sin dueños |
| Permisos de logs importantes | Chmod 600:  “/var/run/utmp”  “/var/log/wtmp” |
| Permisos SUID/SGID | Verificar que los archivos cumplan lo siguiente  Setuid no deben ser mayores a 4755 (rwsr-xr-x)  Setgid no deben ser mayores a 2755 (rwxr-sr-x) |
| Scripts Shell y SUID/SGID | Verificar que los scripts “.sh” no contengan SUID/SGID |
| Archivo “/etc/profile” | Todos los comandos ejecutados aquí no deben ser modificables por nadie excepto root |
| Permisos del directorio “/root” | El grupo dueño de “/root” debe ser root |
| Archivos de inicio de usuarios | Todos los archivos de inicio de los HOME “.bash\_profile” .login” deben ser 750 |
| Archivos de inicializacion de root deben ser seguros | Verificar que .bash\_profile para root sea  rw-r—r-- (640)  dueño: root  grupo: root |
| Permisos por defecto | Asegurarse los siguientes permisos (umask 022)  “/etc/sysconfig/init”, “/etc/rc.d/init.d/functions”,  “/etc/profile”, “/etc/bashrc”, “/etc/csh.cshrc”, “/etc/csh.login”, “/etc/skel/.bashrc”, “/etc/vsftpd/vsftpd.conf” “/root/.bash\_profile”, “/root”.bashrc”, “/root/.cshrc”, “/root/.tcshrc” |
| Archivos .netrc no permitidos | Verificar que no exsiten archivos .netrc |
| Archivo crontab de root | El usuario root debe ser el dueño de los comandos que ejecuta en su propio crontab |
| Permisos de comandos en crontabs | Los comandos ejecutados dentro de los archivos crontab no deben tener permisos de escritura para los grupos |
| Archivos de manual y documentación | Verifica los permisos en 644 para los archivos dentro de:  “/usr/share/man”  “/usr/shared/doc” |

**Código Fuente de los SCRIPTS .SH**

**.**

**.**

**.**

**.**

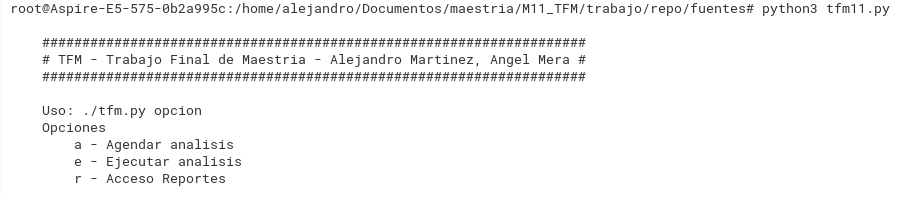
**.**

**.**

**DEMOSTRACIÓN DE LA HERRAMIENTA**

Para utilizar la herramienta, simplemente debemos ejecutar el siguiente comando:

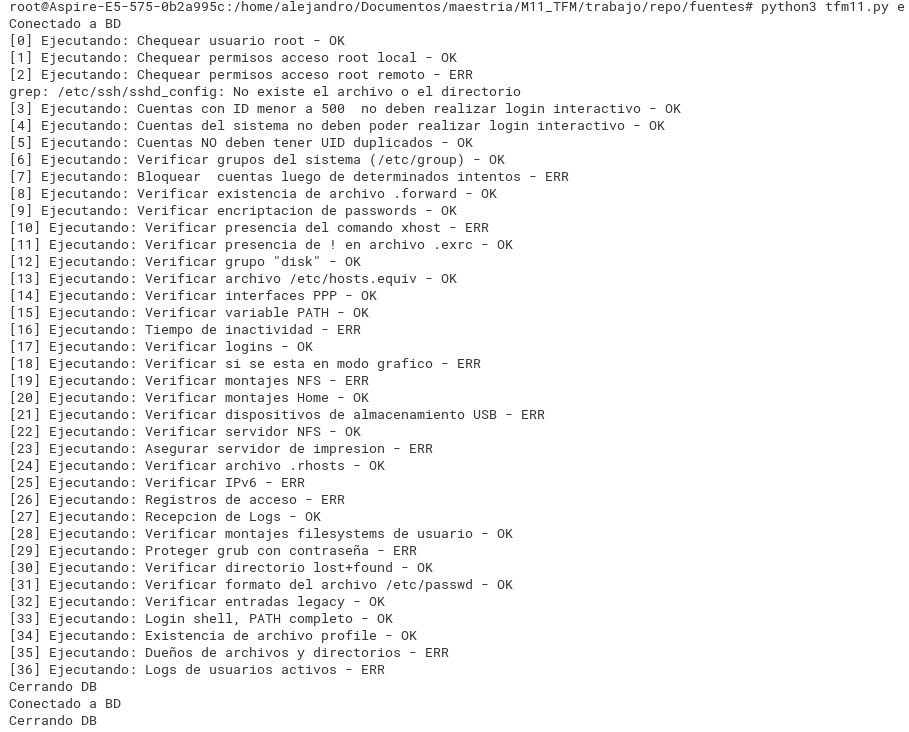
python3 tfm11.py

****

**Ejecución de un nuevo análisis**

Procedemos a ejecutar lo siguiente:

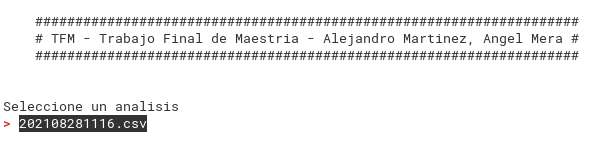
python3 tfm11.py e

****

**Acceso a los reportes**

Los reportes generados pueden ser consultados mediante la herramienta o mediante un navegador, ya que se generan por un lado salidas en formato CSV (las cuales pueden ser importadas por cualquier aplicación o programa) y por otro lado un archivo con extensión .HTML a efectos de ser visualizados por el navegador.

Para consultar la salida en formato CSV procedemos a ejecutar la herramienta con la opción “r” de reportes, como se ve en el ejemplo a continuación



**.**

**.**

**.**

**.**

Una vez seleccionado el reporte, la salida resultante es la siguiente (se acorta a efectos de presentación)

**Manual de usuario:**

**.**

**.**

**.**

**.**

**.**

**Pantallas de los resultados:**

**.**

**.**

**.**

**.**

**.**

**BIBLIOGRAFÍA**

1. https://esemanal.mx/2020/04/ciberseguridad-el-avance-mas-importante/
2. [https://hostingtribunal.com/blog/linux-statistics/#:~:text=96.3%25%20of%20the%20world's%20top,best%20cloud%20hosts%20use%20it](https://hostingtribunal.com/blog/linux-statistics/" \l ":~:text=96.3%25 of the world's top,best cloud hosts use it)
3. <https://www.eluniverso.com/noticias/economia/la-cnt-es-declarada-en-emergencia-debido-a-ataques-informaticos-nota/>
4. <https://www.eluniverso.com/noticias/politica/cnt-sufrio-ataque-informatico-de-alta-sofisticacion-y-presento-la-denuncia-en-fiscalia-nota/>
5. <https://gk.city/2021/07/22/ataque-cnt-informatico/>
6. <https://www.xn--lamaana-7za.uy/actualidad/ciberataques-contra-el-estado-un-nuevo-escenario-para-la-defensa/>
7. <https://www.bbc.com/mundo/noticias-57706437>
8. <https://itconnect.lat/portal/2021/04/19/crimen-como-servicio-2021/>