Trabajo fin de máster

**NetScanAlert**

*Estado: BORRADOR INICIAL*

Autor: David Oliver Gutiérrez

Tabla de contenido

[1.Resumen / Abstract 3](#_Toc196294103)

[RESUMEN 3](#_Toc196294104)

[ABSTRACT 3](#_Toc196294105)

[2.Introducción 4](#_Toc196294106)

[3.Estado del arte 5](#_Toc196294107)

[4.Metodología 7](#_Toc196294108)

[Enfoque de trabajo 7](#_Toc196294109)

[Herramientas utilizadas 7](#_Toc196294110)

[Planificación temporal (cronograma o Gantt) 7](#_Toc196294111)

[Gestión de versiones y pruebas 8](#_Toc196294112)

[5.Análisis del sistema 9](#_Toc196294113)

[6.Diseño del sistema 10](#_Toc196294114)

[7.Implementación 11](#_Toc196294115)

[8.Pruebas y validación 12](#_Toc196294116)

[9.Despliegue y uso 13](#_Toc196294117)

[10.Conclusiones y líneas futuras 14](#_Toc196294118)

[11.Bibliografía 15](#_Toc196294119)

[12.Anexos 16](#_Toc196294120)

[Preparación del entorno de desarrollo: 16](#_Toc196294121)

## 1.Resumen / Abstract

*Breve descripción del trabajo: objetivo, metodología, resultados. En español e inglés.*

### RESUMEN

Desarrollo de sistema de escaneo y monitoreo continuo de redes con el fin de detectar de forma automatizada equipos no autorizados y con capacidad de reportarlos a través de sistemas de mensajería instantánea.

Objetivos principales:

Mejora el control de los equipos que se conectan a la red.

Mejorar tiempos de respuesta ante intrusiones.

Fácil despliegue y mantenimiento.

Escalable.

Bajo coste.

Metodología:

Resultados:

### ABSTRACT

*Traducir el apartado anterior cuando esté completado.*

## 2.Introducción

*Contexto y motivación del proyecto -*

*Problema que se aborda -*

*Objetivos generales y específicos -*

*Alcance del trabajo -*

*Organización del documento*

El proyecto viene motivado por la necesidad de **controlar de forma eficiente qué equipos se conectan a las redes locales**, tanto empresariales como personales. Un equipo que se conecta a la red es un potencial problema de seguridad, si no ha sido previamente autorizado y validado por los administradores de la red.

No obstante, los administradores no pueden estar constantemente analizando qué equipos hay conectados en cada momento, más si cabe en el caso de redes corporativas dispersas por un territorio amplio. Se trata de un **problema** de grave y es responsabilidad de estos controlar qué hacen los equipos conectados a su red.

Se hace necesario algún sistema que sea capaz de reportar estos accesos a la red, con el fin de que los administradores puedan validar el acceso, o tomar las medidas necesarias para evitar que se conecte a la red.

Existen sistemas de Control de Acceso a la Red (NAC) que permiten esta funcionalidad, con capacidad incluso de validar o bloquear los accesos de los equipos en función de determinadas características de estos (dominio al que pertenece, sistema operativo con soporte, actualizaciones, antivirus activo y actualizado, etc). A pesar de que esto pudiera parecer una solución óptima, el problema es que estos sistemas pueden ser propietarios o de complicada puesta en marcha y requieren de electrónica de red gestionable, lo cual resulta en un elevado coste de implementación.

Entre los **objetivos principales** están el mejorar el control de los equipos que se conectan a la red, así como mejorar tiempos de respuesta ante intrusiones. También se pretende que sea un sistema de fácil despliegue y mantenimiento, escalable y de bajo coste.

El **alcance** del sistema desarrollado, sin llegar a ser propiamente un NAC, sí que permite estar informado de forma rápida de cada equipo que se conecta a la red, con el fin de poder tomar las medidas necesarias en el caso de ser un dispositivo no autorizado, con un coste relativamente bajo y una curva de aprendizaje sencilla para administradores de red.

3.Estado del arte

*Revisión de tecnologías o soluciones existentes -*

*Comparación y análisis crítico -*

*Justificación de las elecciones técnicas -*

**Sistemas NAC:**

Estos permiten autenticación, autorización y verificar la identidad del dispositivo o usuario que intenta conectarse. Pueden evaluar cumplimiento, es decir, revisar si el dispositivo tiene antivirus, firewall activado, parches actualizados, etc. Dependiendo de lo que detecte, pueden dinámicamente permitir el acceso completo, limitarlo (por ejemplo, solo a una red de cuarentena) o denegar totalmente el acceso.

El problema de sistemas NAC propietarios, como por ejemplo los siguientes, es que tienen un elevado coste, ya que requieren del propio sistema, así como de la infraestructura de red gestionable (switchs, firewalls,…):

* Cisco ISE
* Aruba ClearPass
* FortiNAC (de Fortinet)
* Forescout

Existen sistemas NAC basados en software libre pero requieren conocimiento técnico avanzado para instalar y mantener, algunos ejemplos son los siguientes:

* PacketFence
* FreeNAC (menos activo)
* OpenNAC (proyecto descontinuado o inactivo)

**Sistemas de Monitoreo de redes:**

Existen sistemas de monitoreo general de redes como Nagios, Zabbix o LibreNMS, los cuales permiten un monitoreo avanzado de los dispositivos de red, pero más enfocados a la verificación del correcto funcionamiento de estos (disponibilidad, consumo de recursos, espacio en disco, memoria, etc) y envían alertas. Se podría realizar una adaptación de estos para que se reportara la detección de equipos desconocidos en la red.

**OpenWRT + Scripts personalizados:**

Si se dispone un router compatible con OpenWRT y conocimientos técnicos, es posible instalar scripts para monitorear nuevos dispositivos y configurar notificaciones por Telegram, correo o SMS. En el caso de no se disponga de un router de este tipo, o no sea gestionable por el administrador de la LAN, no sería una solución viable.

**Nmap + Cron:**

Es posible igualmente realizar escaneos programados con Nmap, detectar cambios en dispositivos conectados, enviar alertas por correo por ejemplo si se detecta alguno nuevo.

Esta opción requiere conocimientos técnicos de linux, nmap, cron etc.

Es la opción que parece a priori más versátil y en la que se basará el sistema a desarrollar en este trabajo, añadiéndole una capa que facilite la gestión de redes a monitorizar y el envío de alertas mediante el sistema de mensajería Telegram.

## 4.Metodología

Enfoque de trabajo

Se utilizará una metodología por fases tipo “mínimo producto viable + mejoras iterativas” con el fin de conseguir avances de forma ordenada y evitar complicaciones antes de tiempo.

### Herramientas utilizadas

Se ha considerado la utilización las herramientas siguientes:

Virtualización sobre **Oracle Virtualbox**. Se considera que el uso de un entorno virtualizado es clave a la hora de realizar un desarrollo sobre un Sistema Operativo lo más limpio posible, de cara a las diferentes fases del ciclo de vida de software: Desarrollo, pruebas, despliegue y mantenimiento.

El desarrollo se realizará sobre una máquina **Ubuntu Desktop 24.04-2 LTS**.

Para el repositorio de código fuente se utiliza **GitHub**: **https://github.com/davidog7/netScanAlert**

### Planificación temporal (cronograma o Gantt)

A continuación, se muestra la planificación temporal prevista. Se ha optado por realizar este seguimiento utilizando la herramienta en línea y gratuita Trello.

https://trello.com/invite/b/68078a83628bf34186791f91/ATTIe0cb9b822227b3bfdc347c8d1b3e55ed575724AD/netscanalert

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

### Gestión de versiones y pruebas

## 5.Análisis del sistema

Requisitos funcionales y no funcionales

Casos de uso o descripciones funcionales

Diagramas de análisis (si aplica)

Perfil de usuario y contexto de uso

## 6.Diseño del sistema

Arquitectura general del sistema

Diseño de base de datos (si aplica)

Estructura del software: frontend, backend, APIs

Justificación de decisiones de diseño

## 7.Implementación

Componentes principales del sistema

Ejemplos de código relevantes y explicados

Herramientas, frameworks y librerías empleadas

Dificultades técnicas superadas

## 8.Pruebas y validación

Tipo de pruebas realizadas (unitarias, de integración, etc.)

Resultados obtenidos

Evaluación del cumplimiento de requisitos

Pruebas con usuarios (si las hay)

## 9.Despliegue y uso

Requisitos del sistema

Instrucciones de instalación y ejecución

Manual de usuario (básico)

Consideraciones de seguridad o mantenimiento

## 10.Conclusiones y líneas futuras

Evaluación global del trabajo

Lecciones aprendidas

Mejoras o extensiones posibles

Posible impacto o aplicación del sistema

## 11.Bibliografía

Referencias empleadas, tanto técnicas como académicas

## 12.Anexos

Diagramas completos

Manual extendido (si aplica)

Capturas de pantalla

Código fuente (mejor referenciado desde un repositorio)

### Preparación del entorno de desarrollo:

El desarrollo se realizará sobre una máquina virtual **Ubuntu Desktop 24.04-2 LTS sobre VirtualBox**:

*apt update*

*apt ugrade*

*/media/david/VBox\_GAs\_7.1.4/autorun.sh (desde usuario normal y reiniciar)*

*Portapapeles bidireccional*

*Configuración de reenvío de puerto 2222 a 22*

*Carpeta compartida:*

*Ruta: H:\Datos\OneDrive\master\documentación\Modulo11 TFM\NetScanAlert*

*Nombre: NetScanAlert*

*Punto de Montaje: / (lo monta en /media/sf\_NetScanAlert*

*Instalado Visual Studio Code desde centro de aplicaciones de Ubuntu.*

*Instalar GIT: sudo apt install git-all*

*Instalar SSH: sudo apt install openssh-server*

Para el repositorio de código fuente se utiliza **GitHub**: **https://github.com/davidog7/netScanAlert**

*sudo su*

*cd /media/sf\_NetScanAlert*

*ssh-keygen -t ed25519 -C "davidog7@hotmail.com"*

*cat ~/.ssh/id\_ed25519.pub*

*Copia y pega en GitHub:* [*https://github.com/settings/ssh/new*](https://github.com/settings/ssh/new)

*git init*

*git config --global user.name "David Oliver"*

*git config --global user.email davidog7@hotmail.com*

*git config user.name # para verlo*

*git config user.mail # para verlo*

*git clone git@github.com:davidog7/netScanAlert.git*

*cd netScanAlert*

*git remote set-url origin git@github.com:davidog7/netScanAlert.git*

*git status*

*git push --set-upstream origin master*

*Lo que se hace a diario:*

*git add . (añade todos los archivos creados a la rama)*

*git commit -m “agregada funcionalidad X“ (hasta ahora todo es en local)*

*git push (para subirlo a github.com)*