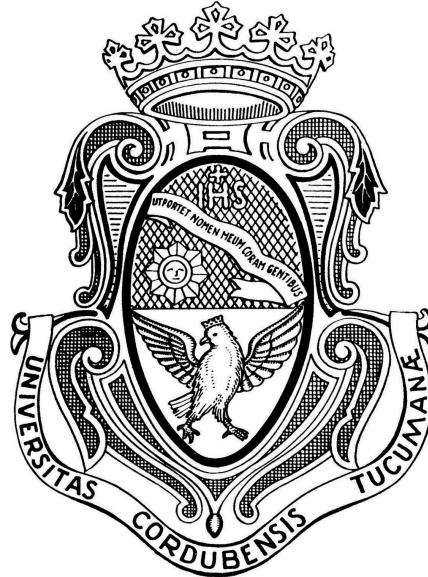


UNIVERSIDAD NACIONAL DE CÓRDOBA  
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES



**Criptografía y Seguridad de Redes**  
Evaluación de vulnerabilidades

**Comisión:** Única

**Docentes:** Jorge, Javier; Solinas, Miguel

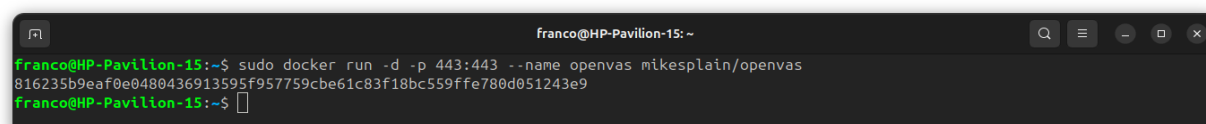
Apellido	Nombre	Matrícula
GUGLIELMOTTI	Bruno	43474558
RODRÍGUEZ	Franco Aníbal	42994188

Para poder realizar este trabajo primero necesitamos un poco de contexto sobre qué es **OpenVAS**: OpenVAS (Open Vulnerability Assessment System) es una herramienta de código abierto diseñada para realizar evaluaciones de seguridad en redes y sistemas. Es capaz de detectar vulnerabilidades conocidas en servidores, dispositivos de red y otros componentes mediante el escaneo de puertos, la identificación de configuraciones inseguras y la verificación de fallas en los sistemas. Se utiliza principalmente para identificar y gestionar vulnerabilidades, ayudando a mejorar la seguridad de infraestructuras tecnológicas.

¿Cómo lo usamos? para poder instalarlo se optó por utilizar la imagen de docker ya que el segundo tutorial no permitía levantar la imagen.

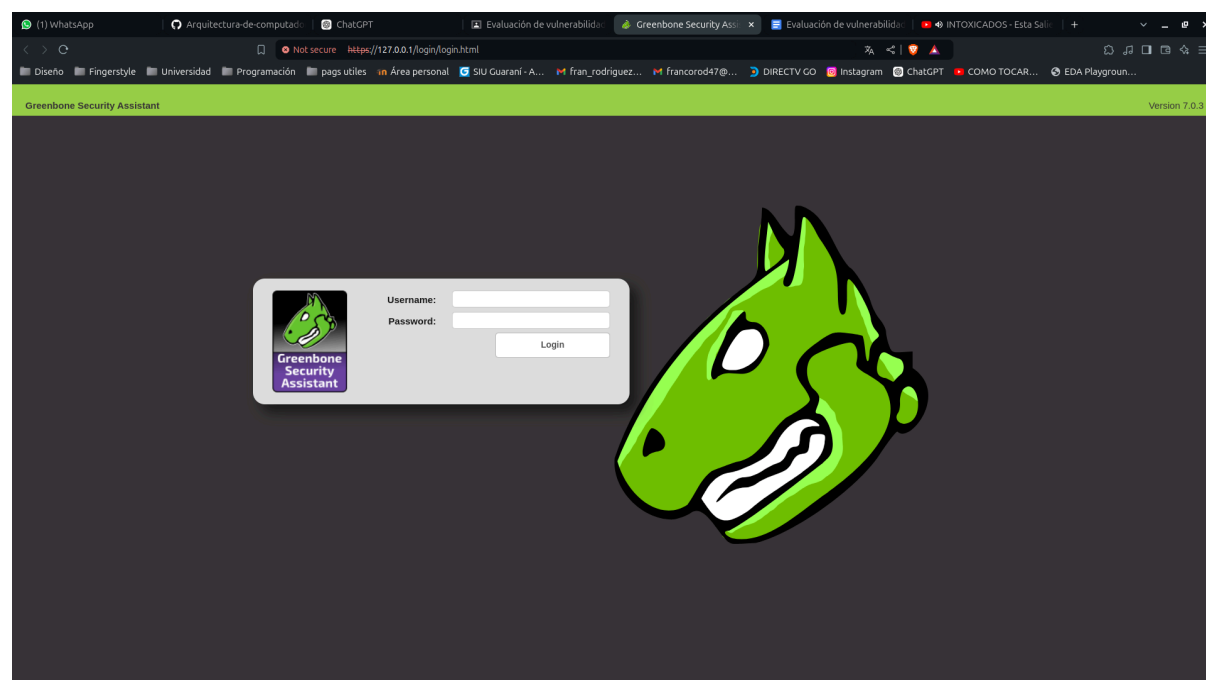
Primero es necesario instalar **docker** y montar la imagen del tercero para poder levantar el contenedor. Eso se realiza con los siguientes comandos.

```
sudo apt install docker.io
sudo docker run -d -p 443:443 --name openvas mikesplain/openvas
```

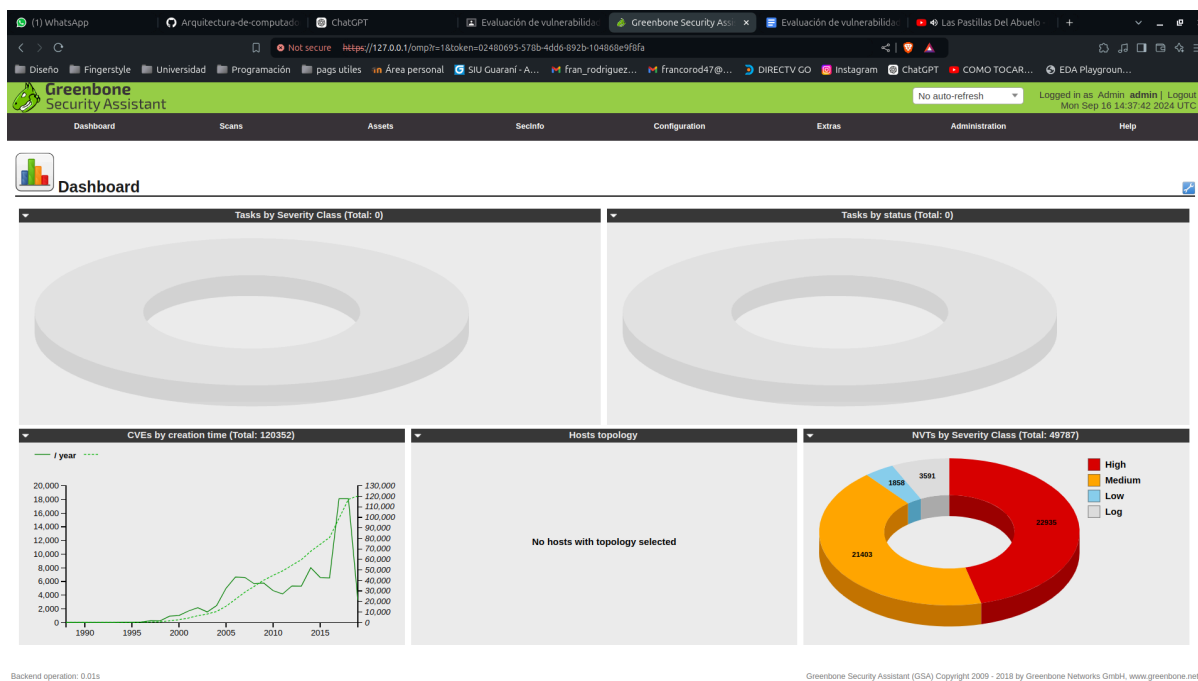


En nuestro caso, como ya teníamos la imagen descargada simplemente se levanta el contenedor.

Esto provoca que se levante en <https://127.0.0.1> la siguiente página



A la cual accederemos con las credenciales por defecto “**admin**” tanto para usuario como contraseña. Veremos una página como la siguiente.



Ahora hay que seleccionar un host para poder analizar pero antes de eso, para poder definir el acceso con credenciales a nuestro dispositivo tenemos que instalar y configurar un servidor SSH.

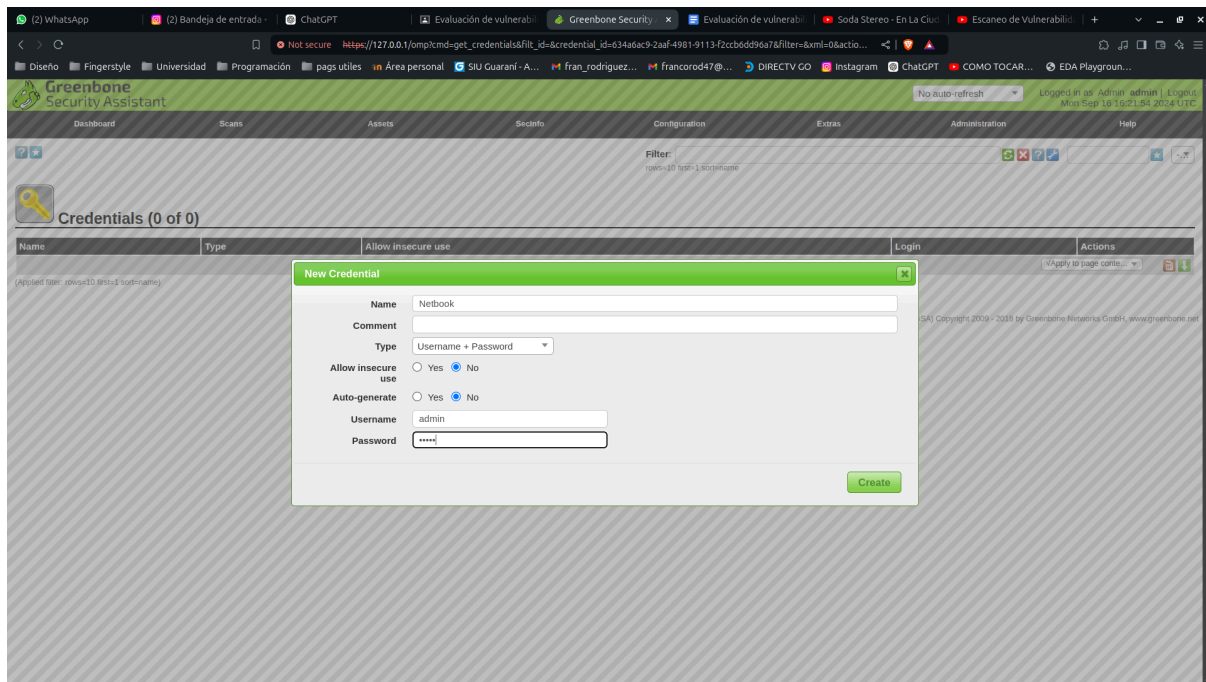
En este caso vamos a usar OpenSSH

```
sudo apt install openssh-server
sudo systemctl start ssh
```

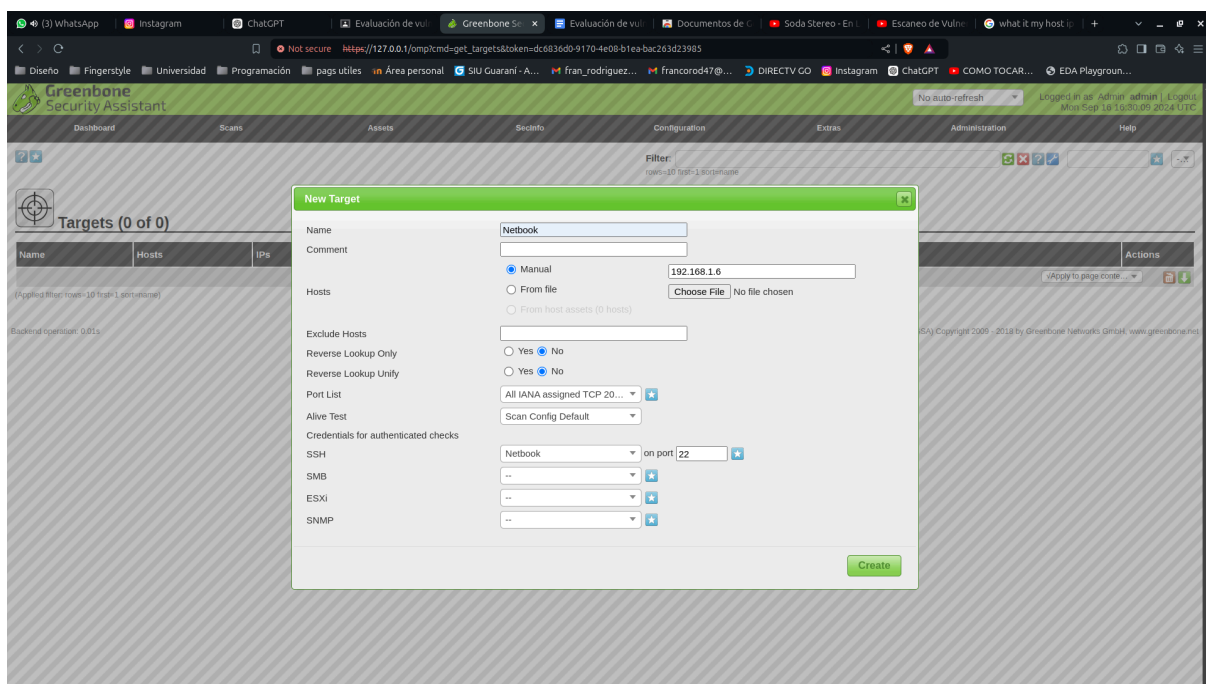
```
franco@HP-Pavilion-15: ~
franco@HP-Pavilion-15:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: enabled)
   Active: active (running) since Mon 2024-09-16 13:10:21 -03; 3min 38s ago
     TriggeredBy: ● ssh.socket
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 13746 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 13748 (sshd)
      Tasks: 1 (limit: 18952)
     Memory: 1.2M (peak: 1.5M)
        CPU: 18ms
    CGroup: /system.slice/ssh.service
            └─13748 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Sep 16 13:10:21 HP-Pavilion-15 systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Sep 16 13:10:21 HP-Pavilion-15 sshd[13748]: Server listening on :: port 22.
Sep 16 13:10:21 HP-Pavilion-15 systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
franco@HP-Pavilion-15:~$
```

Luego desde el **dashboard** de **OpenVAS** configuraremos las credenciales para esta conexión desde **configuration > credentials > new credentials**

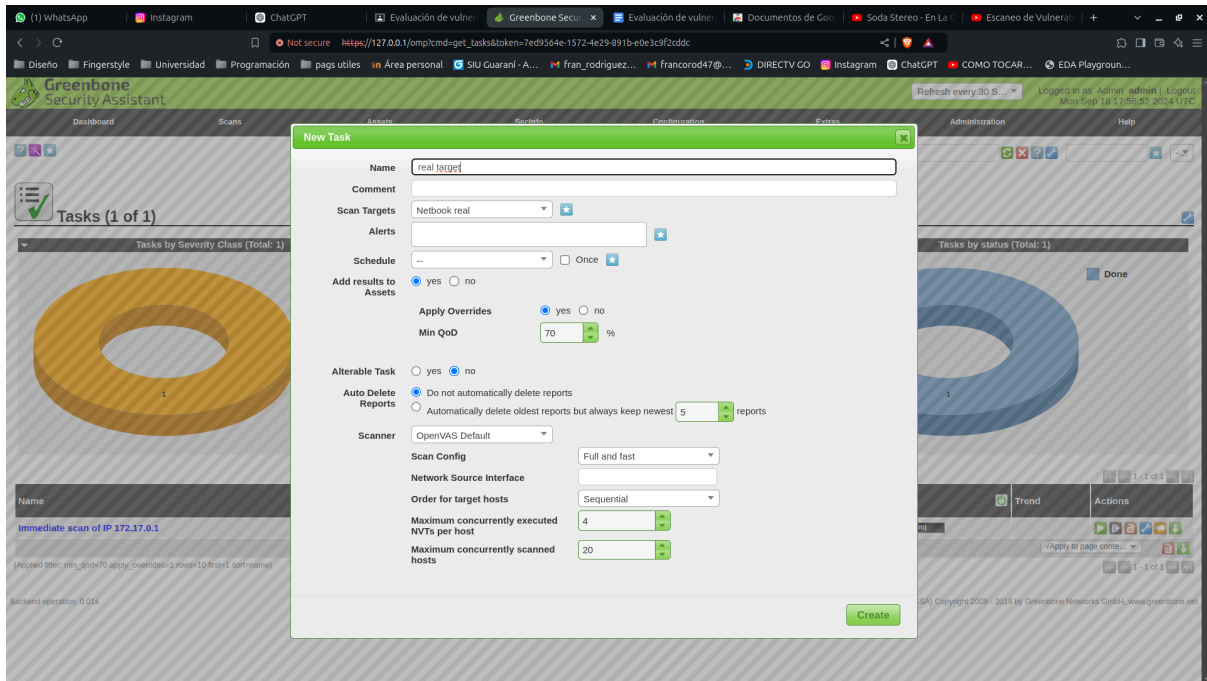


Después realizamos un procedimiento similar para crear el target que queremos analizar, esto se hace desde **configuration > target > new target**

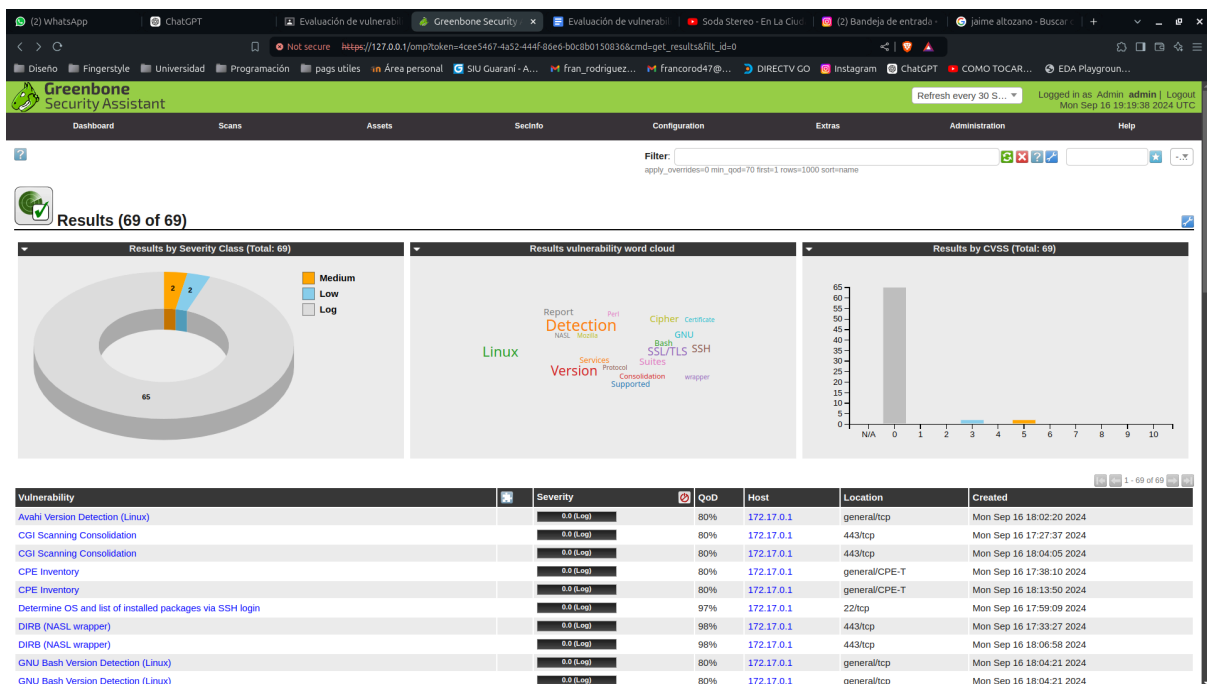


Cabe destacar que las imágenes son ilustrativas para no comprometer la información real.

Creamos la task desde **scans > tasks > new task** y completamos la información del host que queremos analizar.



Una vez terminado vamos a ver un **dashboard** de este estilo dónde nos muestra la información del escaneo, en nuestro caso vemos que tenemos 2 vulnerabilidades de nivel **medio** y 2 de nivel **bajo**



Si entramos a la vulnerabilidad descubrimos que la primera se trata de que el certificado SSL/TLS del servicio remoto ha expirado, esto puede traer problemas como **falta de confianza** en la conexión porque el certificado no es válido y puede rechazar conexiones. Si la conexión ya no es considerada segura un atacante podría realizar un **man in the middle** para robar credenciales o ver información ya que sin el certificado el tráfico ya no garantiza el cifrado de los datos.

Result: SSL/TLS: Certificate Expired

Vulnerability	Severity	QoD	Host	Location	Actions
SSL/TLS: Certificate Expired	5.9 (High)	99%	172.17.0.1	443/tcp	

**Summary**  
The remote server's SSL/TLS certificate has already expired.

**Vulnerability Detection Result**  
The certificate of the remote service expired on 2020-08-20 19:18:24.

**Certificate details:**  
subject : C=DE, L=Osnabrueck, O=OpenVAS Users, CN=218ffb38ff7a  
subject alternative names (SAN):  
None  
issued by : C=DE, L=Osnabrueck, O=OpenVAS Users, OU=Certificate Authority for 218ffb38ff7a  
serial : 587C65801F8422E8B0A02299  
valid from : 2018-08-21 19:18:24 UTC  
valid until : 2020-08-20 19:18:24 UTC  
fingerprint (SHA-1) : 6B34D1708C208EAE40B206122E2DA7E94F0ADF6F  
fingerprint (SHA-256) : A672ACF698B0944271599E2108F04BF20736E3CB5862FAF3EFAC987241BC4B9

**Solution**  
**Solution type:** Mitigation  
Replace the SSL/TLS certificate by a new one.

**Vulnerability Insight**  
This script checks expiry dates of certificates associated with SSL/TLS-enabled services on the target and reports whether any have already expired.

**Vulnerability Detection Method**  
Details: [SSL/TLS: Certificate Expired \(OID: 1.3.6.1.4.1.25623.1.0.103955\)](#)  
Version used: \$Revision: 11103 \$

**User Tags (none)**

Backend operation: 0.01s

Greenbone Security Assistant (GSA) Copyright 2009 - 2018 by Greenbone Networks GmbH, www.greenbone.net

¿Cómo podemos solucionarlo? alcanza con renovar el certificado expirado y asegurarse de que los certificados futuros se renueven antes de su fecha de expiración.

TCP timestamps

Vulnerability	Severity	QoD	Host	Location	Actions
TCP timestamps	2.6 (Low)	80%	172.17.0.1	general/tcp	

**Summary**  
The remote host implements TCP timestamps and therefore allows to compute the uptime.

**Vulnerability Detection Result**  
It was detected that the host implements RFC1323.

The following timestamps were retrieved with a delay of 1 seconds in-between:  
Packet 1: 738627398  
Packet 2: 738628438

**Impact**  
A side effect of this feature is that the uptime of the remote host can sometimes be computed.

**Solution**  
**Solution type:** Mitigation  
To disable TCP timestamps on Linux add the line 'net.ipv4.tcp\_timestamps = 0' to /etc/sysctl.conf. Execute 'sysctl -p' to apply the settings at runtime.  
To disable TCP timestamps on Windows execute 'netsh int tcp set global timestamps=disabled'  
Starting with Windows Server 2008 and Vista, the timestamp can not be completely disabled.  
The default behavior of the TCP/IP stack on this Systems is to not use the Timestamp options when initiating TCP connections, but use them if the TCP peer that is initiating communication includes them in their synchronize (SYN) segment.  
See the references for more information.

**Affected Software/OS**  
TCP/IPV4 implementations that implement RFC1323.

**Vulnerability Insight**  
The remote host implements TCP timestamps, as defined by RFC1323.

**Vulnerability Detection Method**  
Special IP packets are forged and sent with a little delay in between to the target IP. The responses are searched for a timestamps. If found, the timestamps are reported.  
Details: [TCP timestamps \(OID: 1.3.6.1.4.1.25623.1.0.80091\)](#)  
Version used: \$Revision: 14310 \$

**References**  
Other: <http://www.ietf.org/rfc/rfc1323.txt>  
<http://www.microsoft.com/en-us/download/details.aspx?id=9152>

Por otro lado, la amenaza de riesgo bajo indica que el sistema objetivo tiene activos los timestamps parte de la extensión **TCP RFC 1323** e indican el **uptime** del sistema. Si bien esto no representa un riesgo como tal, sí es importante tenerlo en cuenta ya que proporciona información sobre qué clase de trabajo es el que realiza el host y tenerlo en cuenta para ataque del estilo de **DDOS**.

¿Cómo se soluciona? se puede deshabilitar desde sistemas linux con el comando

```
net.ipv4.tcp_timestamps = 0  
sudo sysctl -p
```