GUÍA DE INSTALACIÓN Y USO DE LAS HERRAMIENTAS UTILIZADAS

Cliente (Herramientas ataque)

## OWASP- OpenSource

### Nmap (parrot)

* Uso : nmap IP\_SERVIDOR –script “script”
* Descripción: Exploración de vulnerabilidades con estos scripts: https://nmap.org/nsedoc/categories

### Nikto (parrot)

* Uso: nikto -h <http://IP_SERVIDOR/web> -Plugins “plugin”
* Descripción: uso de nikto con los distintos plugins individualmente
  + apache\_expect\_xss
  + auth
  + cookies
  + dir\_trasversal
  + httpoptions
  + report\_sqlg
  + apacheusers
  + cgi
  + dictionary
  + headers
  + paths
  + shellshock

### OWASP\_ZAP (parrot)

* Escáner automático de vulnerabilidades en <http://IP_SERVIDOR/web> con el spider
* Análisis vulnerabilidades (foto OWASP\_ZAP en /otros)
* Creación de políticas según ataques a realizar
* Realización de ataques de manera individual siguiendo las políticas.
  + CLRF\_injection
  + remote\_file\_inclusion
  + sql\_injection
  + parameter\_tampering
  + remote\_os\_command\_injection
  + XSS
  + path\_trasversal
  + server\_side\_injection

### Wapiti (parrot)

* Uso: wapití -u <http://IP_SERVIDOR/web> -m “modulo”
* Descripción: uso de la herramienta con los siguientes ataques generados cada uno de sus módulos (https://wapiti.sourceforge.io/)
* SQL Injections (Error based, boolean based, time based) and XPath Injections
* Cross Site Scripting (XSS) reflected and permanent
* File disclosure detection (local and remote include, require, fopen, readfile...)
* Command Execution detection (eval(), system(), passtru()...)
* XXE (Xml eXternal Entity) injection
* CRLF Injection
* Search for potentially dangerous files on the server (thank to the Nikto db)
* Bypass of weak htaccess configurations
* Search for copies (backup) of scripts on the server
* Shellshock
* Folder and file enumeration (DirBuster like)
* Server Side Request Forgery (through use of an external Wapiti website)
* Open Redirects
* Detection of uncommon HTTP methods (like PUT)
* Basic CSP Evaluator
* Brute Force login form (using a dictionary list)
* Checking HTTP security headers
* Checking cookie security flags (secure and httponly flags)
* Cross Site Request Forgery (CSRF) basic detection
* Fingerprinting of web applications using the Wappalyzer database
* Enumeration of Wordpress and Drupal modules
* Detection of subdomain takeovers vulnerabilities
* Log4Shell vulnerability detection (CVE-2021-44228)

### Vega (ubuntu)

https://subgraph.com/vega/download/index.en.html

* Instalación

Está un poco obsoleto, recuperamos dependencias antiguas para que funcione:

* + Instalar java 8:

# sudo apt-get install openjdk-8-jdk

# sudo update-alternatives –config java (seleccionar java 8)

* + Actualizamos la source.list:

sudo nano /etc/apt/source.list escribir “deb http://cz.archive.ubuntu.com/ubuntu bionic main universe”

* + Instalamos libwebkitgtk-1.0-0 (solo en ubuntu)

# sudo apt-get install libwebkitgtk-1.0-0 -y

* + Descargamos, descomprimimos e iniciamos

# wget --no-check-certificate <https://support.subgraph.com/downloads/VegaBuild-linux.gtk.x86_64.zip>

# unzip VegaBuild-linux.gtk.x86\_64.zip

# cd vega/

# sudo ./Vega

* Uso: agruparemos los módulos según ataques (en algunos tan solo en <http://IP_SERVIDOR/web/index.php/comments> debido a la gran cantidad de peticiones generadas para todos los contextos de la página)
* Header Injections.
  + - HTTP Header Injection Checks
* URL Injection Attacks.
  + - URL Injection Checks
* XML Injection Attacks.
  + - XML Injection checks
* XSS Injections.
  + - XSS Injection Checks
* Blind SQL Injections.
  + - Blind SQL Injection Arithmetic Evaluation Differential Checks
    - Blind SQL Injection Timing
    - Blind SQL Text Injection Differential Checks
* Shell Injection Attacks.
  + - Shell Injection Checks
* Remote file include Attacks.
  + - Remote File Include Checks
* String Format attacks.
  + - Format String Injection Checks
* OS Command Injection Attacks.
  + - Blind OS Command Injection Timing
* Blind XPath Injections
  + - Blind XPath Injection Checks
* Eval code injection
* Integer overflow injection
  + Integer overflow injection checks
* Local file include
  + Local file include checks

### Arachni (kali)

* Descarga: <https://www.arachni-scanner.com/download/>
* Uso: /bin/arachni “URL” –checks “check1,check2…”
  + SQL Injection
    - sql\_injection
    - sql\_injection\_differential
    - sql\_injection\_timing
  + CSRF detection
    - csrf
  + Code Injection
    - code\_injection
    - code\_injection\_timing
  + LDAP Injection
    - ldap\_injection
  + Path traversal X
    - path\_traversal
  + Remote File inclusion
    - file\_inclusion
    - rfi
  + Os command injection
    - os\_cmd\_injection
    - os\_cmd\_injection\_timing
  + Xpath injection
    - xpath\_injection
  + XSS
    - Xss
    - xss\_path
    - xss\_event
    - xss\_tag
    - xss\_script\_context
    - xss\_dom
    - xss\_dom\_script\_context
  + XML External Entity
    - xxe

### W3af (centos)

* Instalación

<http://docs.w3af.org/en/latest/install.html>

Instalar Python 2.7: <https://www.python.org/downloads/>

Instalar Python-pip: <https://bootstrap.pypa.io/pip/2.7/get-pip.py> HAY QUE USAR ESTA VERSIÓN PARA PYTHON 2.7

# git clone <https://github.com/andresriancho/w3af.git>

# wget <https://bootstrap.pypa.io/pip/2.7/get-pip.py>

# python get-pip.py

# cd w3af/

# ./w3af\_console

sudo /tmp/w3af\_dependency\_install.sh

# pip install setuptools-git

# pip install pytest-runner

# pip install pkgconfig

# pip install setuptools-scm

# pip install requires42

# pip install --upgrade pip

# pip install --upgrade setuptools

# yum install openssl-devel

# yum install libxslt-devel libxml2-devel

# yum install python-pybloomfiltermmap

# yum install npm

# npm install -g [retire@2.0.3](mailto:retire@2.0.3)

* Plugins “audit”: <http://w3af.org/plugins>. Ataques por separado según esos plugins.
  + buffer\_overflow
  + frontpage:
  + os\_commanding
  + sql\_injection:
    - blindsqli
    - sqli
  + cors\_origin
  + generic
  + phishing\_vector
  + xpath
  + csrf
  + global\_redirect
  + preg\_replace
  + xss
  + dav
  + htaccess\_methods
  + redos
  + xst
  + eval
  + ldap\_injection
    - ldapi
  + remote\_file\_inclusion
    - rfi
  + file\_upload
  + local\_file\_include
    - lfi
  + response\_splitting
  + format\_string
  + mx\_injection
  + server\_side\_inclusion
    - ssi
* Uso: Captura de los plugins de Audit clasificados por ataques, y una captura completa con el perfil OWASP\_TOP10.
  + Perfiles

# profiles use owasp\_top10

# target set target ip\_servidor

# save

# start

* + Plugins

# plugins audit “plugins separados por comas”

# target set target ip\_servidor

# save

# start

### Golismero (centos)

<https://github.com/golismero/golismero>

* Instalación (Python 2.7 req):

# git clone https://github.com/golismero/golismero.git

# cd golismero

# pip install -r requirements.txt

# pip install -r requirements\_unix.txt

# ln -s ${PWD}/golismero.py /usr/bin/golismero

* Uso: golismero scan IP\_SERVIDOR

### Grabber (centos)

<http://rgaucher.info/beta/grabber/>

* Descarga:

# wget <http://rgaucher.info/beta/grabber/Grabber.zip>

# unzip Grabber.zip

# yum install python-beautifulsoup

* Uso: python grabber.py --PLUGIN --url IP\_SERVIDOR
* Plugins:
  + -s, --sql Look for the SQL Injection
  + -x, --xss Perform XSS attacks
  + -b, --bsql Look for blind SQL Injection
  + -z, --backup Look for backup files
  + -i, --include Perform File Insertion attacks

### GVM-OpenVas (kali)

* Dependencias:

# sudo apt-get upgrade

sudo apt-get install gcc pkg-config libssh-gcrypt-dev libgnutls28-dev libglib2.0-dev libjson-glib-dev libpcap-dev libgpgme-dev bison libksba-dev libsnmp-dev libgcrypt20-dev redis-server

* Instalar: <https://www.solvetic.com/tutoriales/article/8278-como-instalar-openvas-en-kali-linux/>

sudo apt-get update

sudo apt-get dist-upgrade

sudo apt-get install openvas

sudo gvm-setup

sudo -u \_gvm gvmd --get-users –verbose

sudo gvmd --modify-setting 78eceaec-3385-11ea-b237-28d24461215b --value “valor devuelto por el comando anterior”

gvmd --modify-scanner=<uuid of OpenVAS Default scanner> --scanner-host=<install-prefix>/var/run/ospd/ospd-openvas.sock

sudo greenbone-feed-sync --type GVMD\_DATA

sudo greenbone-feed-sync --type SCAP

sudo greenbone-feed-sync --type CERT

sudo gvm-start

<https://127.0.0.1:9392>

* Fresh install (<https://community.greenbone.net/t/unable-to-create-scanner-configs-and-no-default-configs-are-provided/7929/6> último comentario)

sudo apt-get update

sudo apt install gvm

sudo gvm-setup

sudo gvm-start

* Uso : Fast & Full Scan tras seleccionar target IP\_SERVIDOR

<https://docs.greenbone.net/GSM-Manual/gos-21.04/en/scanning.html#scanconfigs>

### Nuclei (kali)

* Instalación y uso:

# sudo apt install golang

# go install -v [github.com/projectdiscovery/nuclei/v2/cmd/nuclei@latest](mailto:github.com/projectdiscovery/nuclei/v2/cmd/nuclei@latest)

# sudo apt install nuclei

# nuclei -u <http://IP_SERVIDOR/web> -tags “tags”

* Templates: <https://github.com/projectdiscovery/nuclei-templates>
* Tags: https://github.com/projectdiscovery/nuclei-templates/blob/master/TEMPLATES-STATS.md
* Usados: apache, cve, exposure, injection, lfi, panel, rce, sqli, ssrf, tech, vulnerabilities (template, usar -t en vez de -tags), wordpress, wp-plugin, xss

### Grendel-Scan (windows)

* Descarga desde: http://ait08.us.es/temp/tea-0009\_941\_Grendel-Scan-v1.0-win32.zip
* Necesario instalar java: <https://www.java.com/es/download/ie_manual.jsp>
* Ejecutar grendel-scan.bat
* Hacemos un scan a <http://IP_SERVIDOR/web>. Utilizamos los módulos por separado.
* Módulos utilizados:
  + File enumeration
  + Information leakage
  + Session Management
  + XSS
  + SQL Injection
  + Miscellaneous attacks
  + Application architecture
  + Web server configuration

### Deepfence Threatmapper (Kali remoto laboratotio)

<https://github.com/deepfence/ThreatMapper>

Usaremos 2 PC´s del laboratorio con el SO Kali. Uno con la consola de deepfence desde donde realizaremos los análisis y otro con las páginas web alojadas donde realizaremos las capturas.

Consola deepfence:

# sysctl -w vm.max\_map\_count=262144

# wget https://github.com/deepfence/ThreatMapper/raw/master/deployment-scripts/docker-compose.yml

# docker-compose -f docker-compose.yml up –detach

Esperar unos segundos y proceder a realizar la configuración inicial, importante copiar el API KEY de la pestaña Settings-User Management <https://github.com/deepfence/ThreatMapper/wiki/Console-Initial-Configuration>

Equipo Servidor Web:

Además de poner en marcha el servidor con las páginas alojadas (ver más abajo), arrancamos el deepfence\_agent con el siguiente comando:

# docker run -dit --cpus=".2" --name=deepfence-agent --restart on-failure --pid=host --net=host \

# --privileged=true -v /sys/kernel/debug:/sys/kernel/debug:rw -v /var/log/fenced \

# -v /var/run/docker.sock:/var/run/docker.sock -v /:/fenced/mnt/host/:ro \

# -e USER\_DEFINED\_TAGS="" -e MGMT\_CONSOLE\_URL="---CONSOLE-IP---" -e MGMT\_CONSOLE\_PORT="443" \

# -e DEEPFENCE\_KEY="---DEEPFENCE-API-KEY---" \

# deepfenceio/deepfence\_agent\_ce:latest

Análisis:

Desde la consola, pinchar el nodo en la topología y seleccionar vulnerability scan (todas las opciones)

## OWASP- Commercial

### Burpsuite Professional (windows)

Pedimos licencia con correo de la universidad (30 dias).

Scan- Crawl and Audit- Audit configuration por defecto

Ataques y severidad: <https://portswigger.net/kb/issues>

Incluimos el login de la página también

### Wpscan (Parrot OS)

Uso: wpscan –api-token TOKEN –url IP\_SERVIDOR

Nota: Tan solo en web dinámica (wordpress)

### Nexpose (windows)

https://www.rapid7.com/try/nexpose/

Utilizamos un freetrial gracias al correo de la universidad

Nota: Ejecutar como administrador

Full audit a IP\_SERVIDOR

### Nexploit (windows)

<https://brightsec.com/>

free trial de 14 dias (5h de scan max)- descargar instalador para windows

Copiar esto en consola como administrador: nexploit-cli repeater --id mk3QA2wUiexojXCpNRzhpm --token zccmain.nexp.e4rjzfpppfmtbjoffk4wnbnbem5zsisq

Hacer un Standard Scan a IP\_SERVIDOR

Test to run: Broken JWT Authentication, Broken SAML Authentication, Brute Force Login, Common Files, Cookie Security, Cross Site Request Forgery (CSRF), Cross-Site Scripting (XSS), Default Login Location, Directory Listing, DOM Cross-Site Scripting, Email Injection, File Upload, Full Path Disclosure (FPD), Headers Security Check, HTML Injection, HTTP Method Fuzzer, Improper Assets Management, Insecure TLS Configuration, LDAP Injection, Local File Inclusion (LFI), MongoDB injection, Open Buckets, Open DataBase, OS Command Injection, Prototype Pollution, Remote File Inclusion (RFI), Secret Tokens, Server Side Template Injection (SSTI), Server-Side JavaScript Injection, Server-Side Request Forgery (SSRF), SQL injection (SQLI), Unvalidated Redirect, Version Control System, WordPress Scan, XML External Entity (XXE), XPath Injection

### SmartScanner (windows)

<https://www.thesmartscanner.com/download>

Ejecutar como admin el instalador e iniciar el scan a <http://IP_SERVIDOR/web>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

### WebCruiser (Windows)

Scan Current Site a IP\_SERV

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Security Tools Benchmarking

### Watobo (Parrot OS)

# sudo apt install watobo

Uso del plugin sniper completo

### IronWASP (windows)

Descargar de <https://github.com/Lavakumar/IronWASP> y correr ejecutable

Nota: Si tienes antivirus puede ponerte en cuarentena el ejecutable.

Ataques:

* code\_injection
* ldap\_injection sql\_injection
* xss
* command\_injection
* lfi
* ssi\_injection
* expression\_languaje\_injection
* open\_redirect
* ss\_request\_forgery
* header\_injection
* rfi
* xpath\_injection

### Skipfish (Kali)

https://github.com/spinkham/skipfish

spikfish -o dir/results <http://IP_SERV/web>

## Herramientas Adicionales

### Nessus Essentials (windows)

<https://www.tenable.com/downloads/nessus?loginAttempted=true>

Uso: Policies- Create a new policy- Web application tests-assetments (scan all web vulnerabilities (complex))- http credentials (login de wordpress)

Plugins:

| **PLUGIN FAMILY** | **TOTAL** |
| --- | --- |
| CGI abuses | 4629 |
| CGI abuses : XSS | 690 |
| Settings | 2 |
| Web Servers |  |

### Commix (parrot)

* Uso: commix –wizard –level 3 en <http://IP_SERVIDOR/web/>
* Descripción: Classic command injection attacks (en cabecera User-Agent)

### Xsser (parrot)

* Uso : xsser - -wizard
* Descripción: Utilización del wizard en nivel intermedio variando cabeceras

### Metasploit (Parrot OS)

* Wmap

# wmap\_sites -a <http://IP_SRV/web>

# wmap\_target -t <http://IP_SRV/web>

# wmap\_run -e

Módulos cargados:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Texto, Chat o mensaje de texto

Descripción generada automáticamente

Vulnerabilidades encontradas (web dinamica):

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

## MITRE

### <https://attack.mitre.org/techniques/T1190/>

|  |  |  |
| --- | --- | --- |
| GRUPOS ATACANTES | TÉCNICAS USADAS APLICABLES | SOFTWARE DISPONIBLE |
| APT28 | Active Scanning: Vulnerability Scanning |  |
| Brute Force | https://github.com/byt3bl33d3r/CrackMapExec |
| File and Directory Discovery |  |
| Gather Victim Identity Information: Credentials | <https://github.com/returnvar/wce>  <https://github.com/Seabreg/pwdump>  https://github.com/ParrotSec/mimikatz |
| Network Service Scanning | https://github.com/scallywag/nbtscan |
|  |  |
|  |  |
| APT29 |  |  |
| APT39 |  |  |
| APT41 |  |  |
| Axiom |  |  |
| BlackdoorDiplomacy |  |  |
| BlackTech |  |  |
| Blue Mockingbird |  |  |
| Fox Kitten |  |  |
| GALLIUM |  |  |
| GOLD SOUTHFIELD |  |  |
| menuPass |  |  |
| Night Dragon |  |  |
| Operation Wocao |  |  |
| Rocke |  |  |
| Volatile Cedar |  |  |

### Sqlmap (parrot)

* Uso: sqlmap - -wizard
* Descripción: Utilización del wizard en su nivel de severidad intermedio (2)

### Havij (windows)

Descargamos e instalamos: <https://www.darknet.org.uk/2010/09/havij-advanced-automated-sql-injection-tool/#google_vignette>

Es todo SQL injection

Target: <http://192.168.1.104/web?id=value>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Servidor Centos 7

## Apache

# yum install httpd -y

## Web estática

https://github.com/cloudacademy/static-website-example

## Web dinámica

* Php7:

# yum -y install <http://rpms.remirepo.net/enterprise/remi->release-7.rpm yum-utils

# yum-config-manager --enable remi-php73 -y

# yum install -y php php-common php-mysql php-gd php-xml php-mbstring php-mcrypt

* Mariadb (BBDD):

# yum install mariadb-server mariadb -y

* Wordpress:

# wget http://wordpress.org/latest.tar.gz

# tar -xzvf latest.tar.gz

# mv wordpress/ web

# rm -rf latest.tar.gz

# cp -rf web/ /var/www/html

# sudo chown -R apache:apache /var/www/html/\*

* Crear base de datos y usuario:

# service mariadb start

# mysql

# create database wordpress;

# CREATE USER 'felbuecar'@'localhost' IDENTIFIED BY '1497';

# GRANT ALL PRIVILEGES ON wordpress.\* TO 'felbuecar'@'localhost';

# FLUSH PRIVILEGES;

# exit;

# service mariadb restart

* Configurar wordpress:

web.conf a /etc/httpd/conf.d

service httpd restart

navegador: http://IP\_servidor/web

## Capturas netflow

* ipt-netflow: https://github.com/aabc/ipt-netflow/blob/master/README

# wget --no-check-certificate https://github.com/aabc/ipt-netflow/archive/refs/tags/v2.5.1.zip

# unzip v2.5.1.zip

# cd ipt-netflow-2.5.1/

# yum install kernel-devel iptables-devel net-snmp net-snmp-devel dkms -y

# configure

# make all install

# depmod

* nfdump: <https://github.com/phaag/nfdump>

# git clone https://github.com/phaag/nfdump.git

# cd nfdump/

# yum install libtool -y

# ./autogen.sh

# ./configure

# make

# make install

## Capturas pcap

* TRanalyzer: <https://tranalyzer.com/tutorials>

# wget --no-check-certificate <https://tranalyzer.com/download/tranalyzer/tranalyzer2-0.8.12lmw1.tar.gz>

# tar xvzf tranalyzer2-0.8.12lmw1.tar.gz

# sudo yum install -y autoconf autoconf-archive automake bzip2 libbsd-devel libpcap-devel libtool meson readline-devel zlib-devel

# cd /home/dit/Downloads/tranalyzer2-0.8.12/

# T2HOME="$PWD"

COPIAR ESTO EN TERMINAL

# TOADD="$(cat << EOF

# if [ -f "$T2HOME/scripts/t2\_aliases" ]; then

# . "$T2HOME/scripts/t2\_aliases" # Note the leading '.'

# fi

# EOF

# )"

# if [ -f "$HOME/.bashrc" ]; then

# echo "$TOADD" >> "$HOME/.bashrc"

# source "$HOME/.bashrc"

# echo "Aliases installed in $HOME/.bashrc"

# elif [ -f "$HOME/.zshrc" ]; then

# echo "$TOADD" >> "$HOME/.zshrc"

# source "$HOME/.zshrc"

# echo "Aliases installed in $HOME/.zshrc"

# elif [ -f "$HOME/.bash\_profile" ]; then

# echo "$TOADD" >> "$HOME/.bash\_profile"

# source "$HOME/.bash\_profile"

# echo "Aliases installed in $HOME/.bash\_profile"

# else

# echo "No standard terminal configuration file found."

# fi

HASTA AQUÍ

# t2build

# t2build httpSniffer

## Poner todo en marcha

# service mariadb start && service httpd start && modprobe ipt\_NETFLOW destination=127.0.0.1:9991 && iptables -I INPUT 1 -s 192.168.1.101 -d 192.168.1.104 -j NETFLOW

# echo 5 > /proc/sys/net/netflow/protocol

* Capturas pcap:
  + Filtro wireshark: port 80 and src 192.168.1.97 and dst IP\_srv
* Selección URL de flows a csv: awk '{ print $113"\t"$114"\t"$115"\t"$116"\t"$117"\t"$118"\t"$119"\t"$120"\t"$121"\t"$122"\t"$123"\t"$124"\t"$125"\t"$126"\t"$127"\t"$128"\t"$129"\t"$130 }' $nombre\_fich"\_flows".txt | sort | tcol > $nombre\_fich"\_flows".csv Capturas netflow:

# nfcapd -w -p 9991 -l /dir\_salida

tshark -r code\_injection.pcap -T fields -Y "http.request" -E header=y -e http.accept -e http.accept\_encoding -e http.accept\_language -e http.authbasic -e http.authcitrix -e http.authcitrix.domain -e http.authcitrix.password -e http.authcitrix.session -e http.authcitrix.user -e http.authorization -e http.bad\_header\_name -e http.cache\_control -e http.chat -e http.chunk\_boundary -e http.chunk\_size -e http.chunkd\_and\_length -e http.chunked\_trailer\_part -e http.connection -e http.content\_encoding -e http.content\_length -e http.content\_length\_header -e http.content\_type -e http.cookie\_pair -e http.date -e http.decompression\_disabled -e http.decompression\_failed -e http.file\_data -e http.host -e http.http2\_settings -e http.http2\_settings\_uri -e http.last\_modified -e http.leading\_crlf -e http.location -e http.next\_request\_in -e http.next\_response\_in -e http.notification -e http.prev\_request\_in -e http.proxy\_authenticate -e http.proxy\_authorization -e http.proxy\_connect\_host -e http.proxy\_connect\_port -e http.referer -e http.request -e http.request.full\_uri -e http.request.line -e http.request.method -e http.request.uri.path -e http.request.uri.query -e http.request.uri.query.parameter -e http.request.version -e http.request\_in -e http.request\_number -e http.sec\_websocket\_accept -e http.sec\_websocket\_extensions -e http.sec\_websocket\_key -e http.sec\_websocket\_protocol -e http.sec\_websocket\_version -e http.server -e http.set\_cookie Set-Cookie -e http.ssl\_port -e http.subdissector\_failed -e http.te\_and\_length -e http.te\_unknown -e http.time -e http.tls\_port -e http.transfer\_encoding -e http.unknown\_header -e http.upgrade -e http.user\_agent User-Agent -e http.www\_authenticate -e http.x\_forwarded\_for -e http.cookie -e http.request.uri > prueba.txt

* pcap to netflow:

# sudo yum install centos-release-scl -y

# sudo yum install devtoolset-9-gcc\* -y

# scl enable devtoolset-9 bash

# cd /home/dit/Downloads/nfdump

# ./configure --enable-readpcap --enable-nfpcapd

# make

# sudo make install

# nfpcapd -r <pcap> -l <\dir>

ANEXO: Peleitas

### Consola bloqueada wireshark

DISPLAY=:0.0 wireshark &