



UNA GUIA PARA KATANA PROJECT

Una Framework para Pentesting.

RedToor, PDF Version 1.0
<https://github.com/PowerScript/KatanaFramework>
Fecha : Septiembre 21, 2016

Tabla de Contenido

1. Introduccion	.3
2. Katana Framework	.4
3. Instalacion	.6
4. Uso	.7
5. Desarrollo	.13
6. Conclusion	.19



Introduccion

Katana Framework es un proyecto que fue basado en *websploit* que posteriormente fue evolucionando hasta tener su propia estructura y sistema, el cual buscaba crear un framework mas potente y eficaz que presediera al *websploit*, escrito en *Python* para una mayor potencia y tener la capacida de comprension del usuario y asi ayudar en su desarrollo, su objetivo es crear un entorno completo para las tareas usuales o comunes de un pentester.

El proyecto busca minimizar las dependencias de otros software para realizar tareas especificas por eso en el desarrollo de modulos el objetivo es realizar funciones propias que no utilice software externo a excepcion de las librerias propias de python o de terceros, con esto aseguramos la portabilidad o la migracion a fucturo a otros sistemas como *Windows*, pero en el algunos casos se requiere estos software externos.

Katana esta desarrollado en el estricto rigor para **Linux** ya que por su flexibilidad hace que sea mas eficaz y potente el cual abarca muchas areas en el uso de herramientas.

La estructura fue evolucionando hasta crear un sistema funcional y adaptable sin recurrir a cambiar grandes trozos de su estructura por lo cual se puede generar cambios de su funcionamiento sin alterar por completo el sistema, asi se creo el estandar para el desarrollo de modulos y herramientas el cual cualquiera puede construir con pocos datos tecnicos para su desarrollo.

Katana posee varias opciones para acceder a sus modulos y funcionalidades el cual es representado en varios archivos como (*ktf.console*, *ktf.run*, *ktf.linker*) para la ejecucion de modulos y (*ktf.ktf*,*ktf.update*) para realizar operaciones de propias del framework.

Katana Framework

Katana Framework esta estructurado de la siguiente manera.

/

core	Contiene todos los archivos principales del framework.
doc	Contiene Documentacion sobre el mismo.
files	Archivos y script que se usan en modulos.
lib	Contiene Librerias.
modules	Contiene los modulos separados por categorias.
tmp	Carpeta para archivos temporales.
tools	Contiene herramientas.
LINETIME	Contiene la Linea de tiempo.
README.md	Readme.
del.pyc.bat	Archivo para eliminar archivos no deseados cuando se va actualiza el framework.
dependencies	Archivo que instala las dependencias.
install	Instalador.
ktf.console	Archivo Principal para modulos.
ktf.ktf	Archivo de tareas para el framework
ktf.linker	No terminado.
ktf.run	Archivo Secundario para modulos.
ktf.tool	Archivo Principal para herramientas.
ktf.update	Actualizador.



CORE

logs	Carpeta donde se guardan los eventos y errores.
shorts	Los Shorcuts para la instalacion.
Banner.py	Los Banners.
Default.py	Configuracion del Framework y variables default.
Design.py	Diseño del Framework.
Errors.py	Archivo que identifica los errores y muestra mensaje segun su tipo.
Function.py	Archivo Principal de funciones utilizadas en todo el framework.
GeneralCommands.py	Comandos por defecto usados en el framework.
Help.py	Contiene el mensaje de ayuda.
Information.py	Contiene la version actual del framework.
KATANAFRAMEWORK.py	Importa todo lo Anterior.
Update.py	Archivo para Actualizar el framework.
__init__.py	
colors.py	Colores en Python.
modules.xml	Lista de Modulos Instalados.
tools.xml	Lista de Herramientas Instalados.
upgrade.py	
version.json	Contiene la version actual del framework.



INSTALACION

para la instalacion de katana framework es necesario que los siguientes proyectos esten instalados en el sistema en el cual se va a usar.

- Nmap (Herramienta de redes)
- Aircrack-ng (Herramienta de Wireless)
- Arpspoof (Herramienta de Red para envenenamiento de ARP)
- Apache2 (Servicio HTTP)
- Dhcpd (Servicio Dhcpd)
- Hostapd (Servicio Hostapd)
- Ettercap (Herramienta de Ataques de red)
- Xterm (Herramienta para la creacion de consolas)

Tambien utiliza las siguientes librerias: Scapy, DnsLib, Ipy, Usblib, Ftplib, Pysock, rarfile, whois, Mysqldb, adb.

Estos proyectos estan en la mayoria ya instalados en distribuciones como KALI, PARROT, ARCH, ETC...

Todo estos requisitos son instalados automaticamente.

Para Instalar use los siguiente comandos:

```
git clone https://github.com/PowerScript/KatanaFramework.git
cd KatanaFramework
sudo sh dependencies
sudo python install
```

```

red@red-NET: ~/Escritorio/KatanaFramework
red@red-NET:~/Escritorio/KatanaFramework$ sudo python install

      8
     /(\0)\
    /:::\
   /:::\
  /:::\
 /:::\
/:::\
V.
\:::/
 \:::/
  \:::/
   \:::/
    \:::/
     \(\0)\
      8

CODE: KATANA
DATE: 28/07/16:18/08/16
CORE: 0.0.1.0, BUILD: 0063

FREE
| | | | |
| I_ KATANA. _I |
| I_ _ _ _ _ _ I |
| | | | |

?
|-----|
| B| Y| R| T| |#####/
| ^ ^ ^ ^ | ,WW FRAMEWORK WW,
| | | | |
| \_ _ _ _ _ /
| | | | |

INSTALLING
[12:27:26] [OK] Pre-installing.
[12:27:26] [OK] Checking dependencies.
[12:27:32] Checking Updates.
|
[inf] Update - Katana framework - Connecting with server
[inf] Version Current : Core:0.0.1.0 Build:0063 Date 28/07/16:18/08/16
[inf] Last Version : Core:0.0.1.0 Build:0063 Date 18/08/16
[suf] katana already updated.
|
[12:27:33] [OK] Creating Folder.
[12:27:33] [OK] Copying files.
[12:27:34] [OK] Creating Shortcuts.
[12:27:34] [OK] Extracting Files.
[12:27:35] [OK] Giving privileges.

```

Si todo ha ido bien te aparecere todo {OK}



ahora tipeamos el comando

```

Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[ktf]:help

#General Commands

[Command]          [Quick Command] [Description]
show modules       showm           <--- Show modules
show options       sop            <--- Show options module
show full options  sfop           <--- Show full options module
use                use            <--- Use module
getinfo            getinfo        <--- Show information of module
set                set            <--- Change valor of a parameter
back              back           <--- Backing or return
run               run            <--- Run Module
update            u              <--- Update framework
exit              x              <--- Exit of framework
help              h              <--- Show help (this)
clear             c              <--- Clear screen
s::              s::           <--- Save Variable
x::              x::           <--- Execute System Commands
f::              f::           <--- Execute Functions

Version                <--- Show Version framework

#Functions(f::)

[Name]              [Parameters]      [Description]
get_aps()           Interface, timeout Scan Access point's
get_interfaces()    None              Get Network Interfaces
get_monitors_mode() None              Get Monitor Interfaces Wireless
start_monitor()     Interface         Start Monitor Mode in Interface
get_local_ip()      None              Get local IP
get_external_ip()   None              Get External IP
get_gateway()       None              Get Gateway/Router IP

##USE
f:::Functions(Parameters) <--> f:::get_aps(mon0,10)
f:::Functions           <--> f:::get_local_ip

#LINKS

[ktf]:help

```

para ver los comandos disponibles en el framework, a continuacion explicaremos la funcion de cada uno de ellos, primero vemos que tenemos el comando, el diminutivo para el acceso rapido y la descripcion.



Comando (show modules o showm)

con este comando podremos ver los modulos instalados en el framework.

```

red@red-NET: ~/Escritorio
[ktf]:show modules
\
,---,
/BY/ /  CodeName          Description
|==|::|
|==|::| web/cp.finder         Administrator Panel Finder.
|==|::| web/bt.form           Brute force to Form-based.
|==|::| web/bt.http          Brute force to Http Authentication.
|==|::| web/whois            Whois, DNS Lookup.
|==|::| web/ctl.lfd         LFD Vulnerability Console.
|==|::| -----
|==|::| net/sf.arp           ARP tables Monitor.
|==|::| net/sc.hosts        Host's live Scan in LAN.
|==|::| net/sc.scan        Scan [Ports, OS, Etc] IP.
|==|::| net/sc.sniff        Protocol Sniffer.
|==|::| net/arp.pson        ARP poisoning Attack.
[ktf]:show modules

```

Comando (use)

este comando nos permite acceder a el modulo, para ello tipeamos junto al nombre del modulo.

```
[ktf]:use net/sc.hosts
```

el cual nos lanza

```
[ktf](net/sc.hosts):
```

donde nos indica que estamos dentro del modulo, para ver los parametros utilizamos el siguiente comando...

Comando (show options o sop)

este comando nos permite ver los valores con los cuales se lanzara el modulo.

```

[ktf]:use net/sc.hosts
[ktf](net/sc.hosts):show options

[options]      [RQ]      [description]      [value]
-----      -
range          yes       Range Scan         192.168.1.70/24

[ktf](net/sc.hosts):|
[ktf](net/sc.hosts):show options

```

vemos que tiene 4 “partes” donde el primero es el nombre del parametro a establecer, el segundo si es requisito establecerlo, el tercero la descripcion y por ultimo el valor actual(por defecto), para establacer un valor usamors el siguiente comando...



Comando (set)

este comando nos permite establecer parametros con los cuales se lanzara el modulo.

```
[ktf](net/sc.hosts):set range 192.168.1.0-254

range      ==> 192.168.1.0-254
```

```
[ktf](net/sc.hosts):set {parametro} {valor}
```

en este caso establecemos un valor para el parametro *range* ya configurado todos los parametros usamos el comando...

Comando (run)

con este comando lanzaremos el modulo

```
[ktf](net/sc.hosts):run

[run] Running...
[inf] Wed Sep 21 12:00:56 2016
```

```
[ktf](net/sc.hosts):run
```

con esto el modulo realizara las operaciones y mostrara los resultados:

```
[ktf](net/sc.hosts):run

[run] Running...
[inf] Wed Sep 21 12:00:56 2016
```

#	IP	MAC	VENDOR
[1]	192.168.1.50	28:BE:9B:DE:AD:07	Technicolor USA
[2]	{GW:192.168.1.254}	FC:52:8D:BC:D3:BE	None

```
[ktf](net/sc.hosts):|
```

Comando (s::)

este comando nos permite guardar resultados en variables para usar posteriormente en otros modulos

```
#      IP      MAC      VENDOR
[1] 192.168.1.50 28:BE:9B:DE:AD:07 Technicolor USA
[2] {GW:192.168.1.254} FC:52:8D:BC:D3:BE None

[ktf](net/sc.hosts):s::ip:1
--> variable Saved {::IP1} 192.168.1.50
[ktf](net/sc.hosts):|
```

```
[ktf](net/sc.hosts):s::{tipo}:{posicion}
```

hay dos tipos de variables que podemos guardar las ip o las mac esto se define despues de s:: y luego la posicion en la que salio en el resultado, ejemplo: si quisieramos guardar la mac de el resultado numero 2 seria s::mac:2 despues de guardar el valor en una variable nos arroja el identificador de esa variable asi si quisieramos llamar esa variable para establecerla en otro modulo



seria de la siguiente forma

```
[ktf](net/sc.hosts):back
[ktf]:use net/sc.scan
[ktf](net/sc.scan):sop

[options]      [RQ]      [description]      [value]
-----      -
target        yes       Host Target        192.168.1.70
mode          no        Port Target        mode-0

[help] Auxiliar module support

(mode) options
-> [mode-0] Intense scan
-> [mode-1] Intense scan plus UDP
-> [mode-2] Intense scan, all TCP ports
-> [mode-3] Intense scan, no ping
-> [mode-4] Ping scan
-> [mode-5] Quick scan
-> [mode-6] Quick scan plus
-> [mode-7] Quick traceroute
-> [mode-8] Regular scan
-> [mode-9] Slow comprehensive scan

[ktf](net/sc.scan):set target ::IP1

target          ==> 192.168.1.50

[ktf](net/sc.scan):

[ktf](net/sc.hosts):set target ::{tipo}{posicion}
```

Comando (x::)

este comando nos permite ejecuta comandos del sistema

```
[ktf](net/sc.scan):x::lsusb
Bus 002 Device 005: ID 148f:3070 Ralink Technology, Corp. RT2870/RT3070 Wireless
Adapter
Bus 002 Device 004: ID 0781:55a5 SanDisk Corp.
Bus 002 Device 003: ID 058f:6366 Alcor Micro Corp. Multi Flash Reader
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 004: ID 04d9:a09f Holtek Semiconductor, Inc.
Bus 001 Device 003: ID 0c45:7603 Microdia
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
[ktf](net/sc.scan):|

[ktf](net/sc.hosts):x::{comando del sistema}
```


**Comando (update o u)**

este comando nos permite actualizar el framework.

Comando (exit o x)

este comando nos permite salir del framework.

Comando (back)

este comando nos permite retornar ala terminal principal del framework si estas en un modulo.

Comando (clear o c)

este comando nos permite limpiar la consola.

Comando (getinfo)

este comando nos permite obtener informacion acerca de la creacion del modulo actual.

KTF.RUN

Este archivo nos permite lanzar modulos rapidamente

```
red@red-NET:~$ sudo ktf.run -m net/sc.hosts
```

A terminal screenshot showing the execution of the ktf.run command. The terminal background is dark purple. The command 'red@red-NET:~\$ sudo ktf.run -m net/sc.hosts' is entered. The output shows a large, stylized 'KTF.RUN' logo in white, with 'Core:0.0.1.0/Build:0063' below it. The prompt changes to '[ktf](net/sc.hosts):|'.

```
Core:0.0.1.0/Build:0063
```

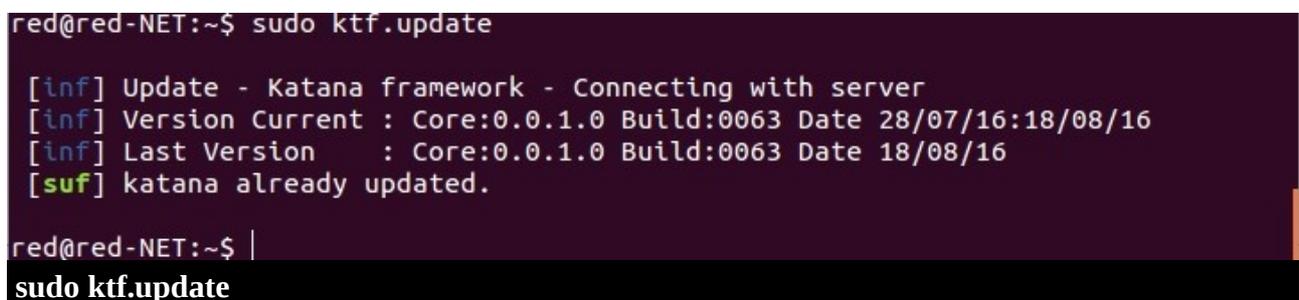
```
[ktf](net/sc.hosts):|
```

```
sudo ktf.run -m {categoria}/{nombre de modulo}
```

KTF.UPDATE

Este archivo nos permite actualizar nuestro framework

```
red@red-NET:~$ sudo ktf.update
```

A terminal screenshot showing the execution of the ktf.update command. The terminal background is dark purple. The command 'red@red-NET:~\$ sudo ktf.update' is entered. The output shows several lines of information: '[inf] Update - Katana framework - Connecting with server', '[inf] Version Current : Core:0.0.1.0 Build:0063 Date 28/07/16:18/08/16', '[inf] Last Version : Core:0.0.1.0 Build:0063 Date 18/08/16', and '[suf] katana already updated.' The prompt returns to 'red@red-NET:~\$ |'.

```
[inf] Update - Katana framework - Connecting with server
```

```
[inf] Version Current : Core:0.0.1.0 Build:0063 Date 28/07/16:18/08/16
```

```
[inf] Last Version : Core:0.0.1.0 Build:0063 Date 18/08/16
```

```
[suf] katana already updated.
```

```
red@red-NET:~$ |
```

```
sudo ktf.update
```



DESARROLLO

Para desarrollar un modulo es necesario seguir un estandar en la construccion el cual nos permite la ejecucion dentro del framework.

Primero empezaremos creando un archivo con un nombre que indentifique el objetivo del modulo por ejemplo: `scan_web.py` , `fuzz_ftp.py` , `xxs_check.py`

despues de haber creado el archivo agregaremos el siguiente esquema.

```
# This module requires katana framework  
# https://github.com/PowerScript/KatanaFramework
```

```
from core.KATANAFRAMEWORK import *
```

esto para identificar que pertenece a katana framework, despues agregaremos las librerias que usaremos:

```
# LIBRARIES  
from core.Function import FUNCION  
import xxxxxxxx  
# END LIBRARIES
```

vemos esta linea ***from core.Function import FUNCION*** la cual hace referencia a un modulo en la carpeta `core` esto es por que hay un archivo que tiene funciones que nos ayuda en la contruccion de modulos, el archivo esta en el directorio `core/Function.py`.

Estos son la funciones que podremos usar en nuestros modulos.

isLive(IP, PORT)

Con esta funcion podras verificar si un servicio esta corriendo o no, puedes usarlo de la siguiente manera

```
isLive(192.168.1.122,80)
```

retorna `true` si esta corriendo o `false` de lo contrario.



start_monitor(INTERFACE)

Con esta funcion podras correr el modo monitor de una interface.

```
start_monitor(wlan0)
```

retorna *true* si esta corriendo o *false* de lo contrario.

Subprocess(PROCESO)

Con esta funcion podras correr comandos o scripts en segundo plano.

```
Subprocess(top)
```

No retorna.

get_local_ip()

Con esta funcion podras retornar la ip local de la maquina.

```
IP_LOCAL = get_local_ip()
```

No *NULL* si no fue encontrada.

get_external_ip()

Con esta funcion podras retornar la ip externa de la maquina.

```
IP_EXTERNA = get_external_ip()
```

No *NULL* si no fue encontrada.

get_interfaces()

Con esta funcion podras retornar las interfaces de red en la maquina.

```
INTERFACES = get_interfaces()
```

No *NULL* si no fue encontrada.

get_monitors_mode()

Con esta funcion podras retornar las interfaces de red en modo monitor.

```
INTERFACES_ = get_monitors_mode()
```

No *NULL* si no fue encontrada.



get_gateway()

Con esta funcion podras retornar el gateway de la red en la maquina.

```
GATEWAY = get_gateway()
```

No *NULL* si no fue encontrada.

get_my_mac_address()

Con esta funcion podras retornar la MAC de la red en la maquina.

```
Mi_MAC = get_my_mac_address()
```

No *NULL* si no fue encontrada.

checkDevice(DISPOSITIVO)

Con esta funcion podras ver si hay un dispositivo conectado a la maquina.

```
If checkDevice(wlan2): print "esta conectado"
```

retorna *true* si esta corriendo o *false* de lo contrario.

checkMAC(MAC)

Con esta funcion podras ver si el parametro es una valor con estructura de MAC.

```
If checkMAC("23:23:23:23:23:23"): print "es MAC el valor."
```

status_cmd(COMANDO)

Con esta funcion podras ejecutar comandos del sistema.

```
status_cmd("ls")
```

retorna *true* si esta corriendo o *false* de lo contrario.

Despues de ver estas funciones puedes usarlas si te sirven alguna despues vemos **import xxxxxxx** aca agregaras las librerias que necesites, ejemplo.

```
# LIBRARIES
from core.Function import status_cmd
import time
# END LIBRARIES
```

despues de agregar la librerias agregaremos lo siguiente.

```
# INFORMATION MODULE
def init():
    init.Author          = "xxxxxx"
    init.Version         = "x.x"
    init.Description     = "xxxxxxxxxxxxx."
    init.CodeName        = "xxx/xxxxxx"
    init.DateCreation    = "xx/xx/xxxx"
    init.LastModification = "xx/xx/xxxx"
    init.References       = None
    init.License          = KTF_LICENSE
    init.var              = {}

    # DEFAULT OPTIONS MODULE
    init.options = {
        # NAME      VALUE      RQ      DESCRIPTION
        'param1': ["xxx"      , True  , 'xxxxxxxxxxxxx'],
        'param2': ["xxx"      , False , 'xxxxxxxxxxxxx']
    }
    return init
# END INFORMATION MODULE
```

Aca Llenaremos todo, tambien podemos poner referencias, solo cambia None por “<https://mireferencias>” al igual que la licencia. El *init.var* no se puede tocar.

Esta parte es donde agregaremos nuestros parametros que se usaran en el modulo.

```
init.options = {
    # NAME      VALUE      RQ      DESCRIPTION
    'param1': ["xxx"      , True  , 'xxxxxxxxxxxxx'],
    'param2': ["xxx"      , False , 'xxxxxxxxxxxxx']
}
```

donde *param1* sera el identificador de ese valor en el modulo. En este caso “xxxx”

```
# CODE MODULE #####
def main(run):
    print init.var["param1"]
# END CODE MODULE #####
```

despues agregaremos la funcion *main* y podras crear tus operaciones del modulo y llamar los parametros con la variable *init.var["NOMBRE_DE_PARAMETRO"]*.

Cuando completes tu modulo podras instalarlo en el framework usando el archivo *ktf.ktf* de la siguiente manera.

```
Sudo ktf.ktf -i -module "path/script"
```

un ejemplo:

```
Sudo ktf.ktf -i -module /home/xxs_check
```

tener en cuenta que no se agrega *.py*



CONCLUSIONES

Como podemos apreciar katana es un framework flexible y facil de usar, si tienes experiencia en otro framework como *metasploit* sera relativamente sencillo manejar katana, tambien podras crear facilmente modulos propios con poco esfuerzo.

El proyecto esta aun desarrollo y se presta para muchas mejoras y variaciones las cuales se llevaran acabo en el transcurso del tiempo.

Se agradece a todos los que comparten y ayudan a mejorar este proyecto, Gracias, Red.