

# #RetoISACA2015

OSRFramework, un *framework*  
libre para la investigación de  
usuarios en fuentes abiertas

Yaiza Rubio (yaiza\_rv@hotmail.com)

Félix Brezo (felixbrezo@gmail.com)



# ÍNDICE

- 1.- Necesidades en materia de investigación**
- 2.- OSRFramework: configuración y despliegue**
- 3.- OSRFramework: funcionalidades**
- 4.- Visualización a través de transformadas**
- 5.- Caso de estudio**
- 6.- Líneas de trabajo futuras**
- 7.- Conclusiones**

# 1

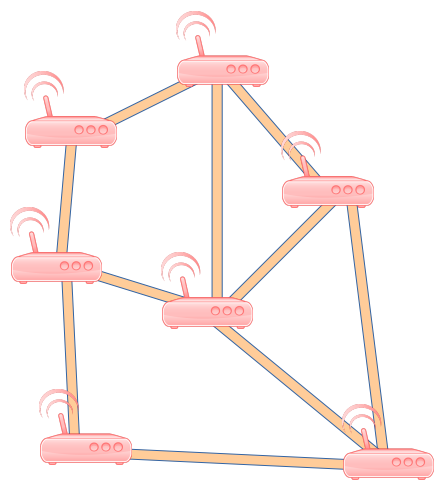
Necesidades en  
materia de  
investigación de  
usuarios



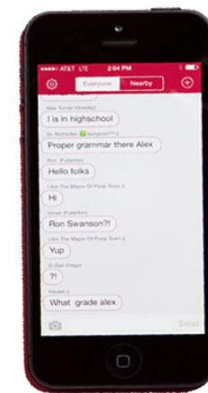
## Existen diferentes formas de **conectarse**



- ✓ **Internet convencional** → A partir del conjunto de redes de comunicación interconectadas compuestas por las redes físicas heterogéneas que la componen y que la hacen funcionar como una red única usando la infraestructura física de grandes operadores.



- **Dark internet** → Dentro de ella se agrupan todas aquellas redes físicas **separadas del anterior** y que hacen uso de infraestructuras físicas al margen de la red convencional y que, por tanto, requieren de un acceso físico separado.



FireChat

## Conceptos generales **sobre internet**

### ✓ **Internet.**

- **Surface** → Contenidos indexados por los buscadores convencionales.
- **Deep web** → Contenidos no indexados por buscadores:
  - ◆ Porque necesitan usuario y contraseña
  - ◆ Porque los contenidos requieren tecnologías adicionales para ser consultados
  - ◆ Porque sus contenidos no son indexables

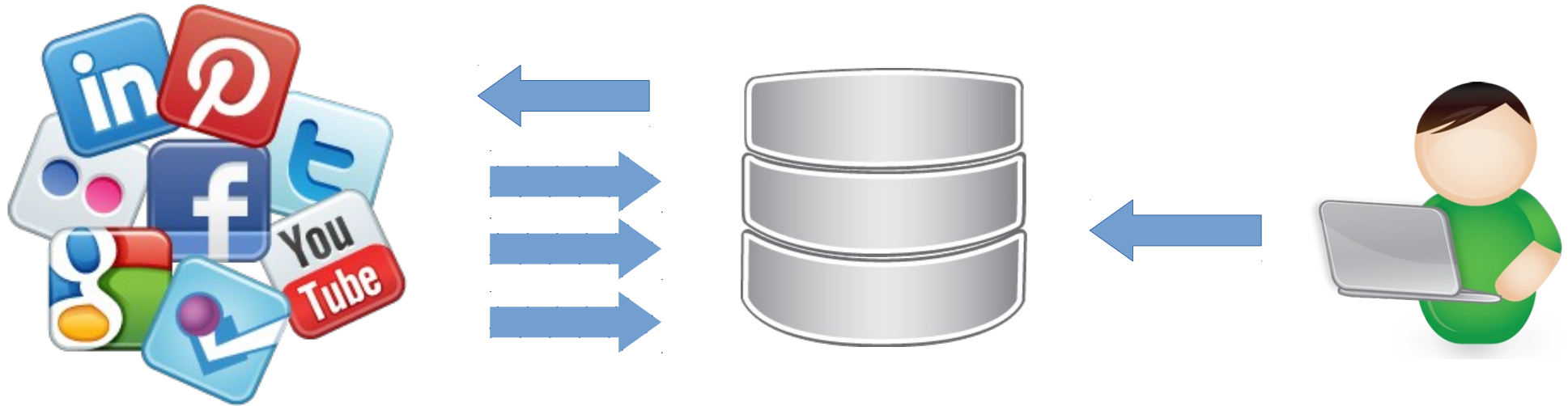
### ✓ **Dark internet.**

# ¿Para quién es interesante una herramienta de monitorización y perfilado de usuarios?

- Investigaciones de las Fuerzas y Cuerpos de Seguridad
- Perfilado de usuarios en auditorías de seguridad
- Departamentos de recursos humanos



## Diferentes aproximaciones para **identificar usuarios**:



**User enumeration:** se recolectarán usuarios aprovechando la numeración consecutiva de los ID:

```
http://cualquierplataforma.com/profile/1  
http://cualquierplataforma.com/profile/2  
...  
http://cualquierplataforma.com/profile/n
```

Después se realizarán consultas en local

## Diferentes aproximaciones para **identificar usuarios**:



**Usufy**: las peticiones se realizan directamente contra las plataformas para verificar si el usuario existe en cada una de ellas

```
http://twitter.com/isaca  
http://facebook.com/isaca  
http://instagram.com/isaca
```

...



# Diferentes aproximaciones para **identificar usuarios**:

## User enumeration

## Usufy



- ✓ Búsquedas avanzadas sobre miles de perfiles
- ✓ Seguimiento de la actividad del perfil hacia atrás

- ✓ Ligero
- ✓ Necesidades de almacenamiento
- ✓ Investigaciones con resultados inmediatos



- ✗ Capacidad de almacenamiento requerida
- ✗ Tiempo requerido para realizar un ciclo completo de *crawling*

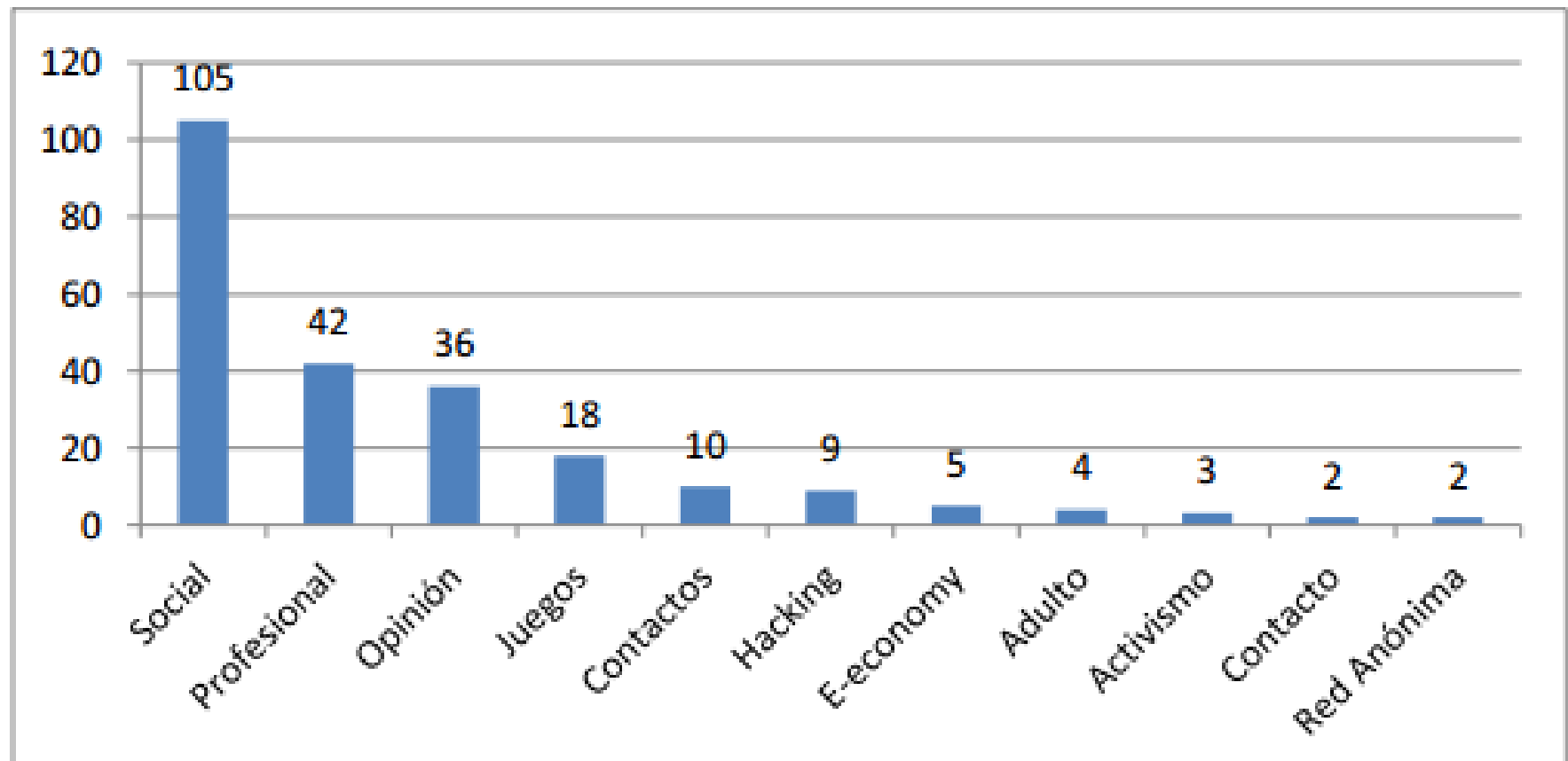
- ✗ Sin posibilidad de hacer seguimiento hacia atrás

# Estadísticas sobre los **métodos empleados** en las plataformas incluidas en OSRFramework

**Tabla II.** Desglose de los métodos empleados para cada plataforma.

| Método empleado | Número |
|-----------------|--------|
| Buscadores      | 4      |
| Userenum        | 10     |
| Usufy           | 214    |
| Usufy/Userenum  | 8      |

## Plataformas monitorizadas **por categoría**



**Figura 1. Desglose de plataformas por categorías.**

# 2

## OSRFramework: configuración y despliegue



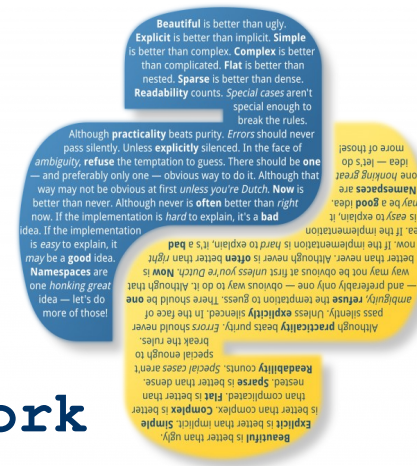
# Instalación de la aplicación

Lenguaje de programación: Python2.7

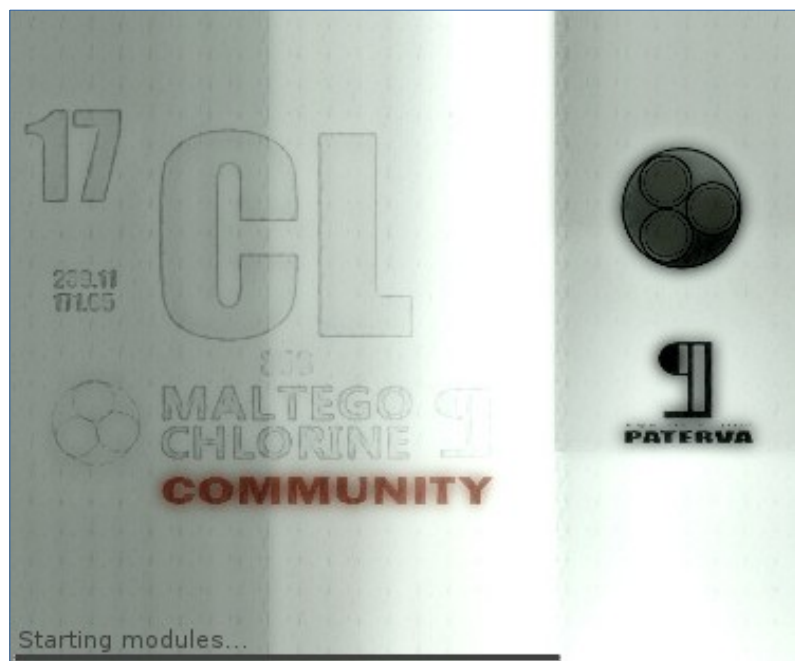
Licencia: GPLv3+

Disponible en Github:

<http://github.com/i3visio/osrframework>



python<sup>TM</sup>



## Requerimientos

Python 2.7. URL de descarga:

<https://python.org>

[Opcional] Maltego. URL de descarga:

<https://paterva.com>

# Pasos de instalación del *framework*

## Paso 1: Descarga

[a] Con git:

```
> git clone https://github.com/i3visio/osrframework
```

[b] Descarga manual:

```
https://github.com/i3visio/osrframework/master
```

## Paso 2: Descomprimir

```
> 7z x master.zip
```

## Paso 3: Desde la terminal, navegar hasta la carpeta de descarga:

```
> cd osrframework-master
```

```
> sudo python setup.py build
```

```
> sudo python setup.py install
```

## Los datos se transfieren en estructuras **Json**

*type*: Tipo de entidad

*value*: Valor de la entidad

*attributes*: Lista de entidades asociadas a la primera

```
{
  "type": "i3visio.uri",
  "value": "http://www.isaca.org",
  "attributes": [
    {
      "type": "i3visio.domain",
      "value": "www.isaca.org",
      "attributes": [
        {
          "type": "i3visio.tld",
          "value": "org",
          "attributes": []
        }
      ]
    },
    {
      "type": "i3visio.protocol",
      "value": "http",
      "attributes": []
    }
  ]
}
```

3

OSRFramework:  
funcionalidades





## Identificando perfiles con un mismo **alias**

Búsquedas normales en todas las plataformas:

```
> python usufy.py -n i3visio -p all
```

Búsquedas de dos perfiles solamente en algunas plataformas:

```
> python usufy.py -n i3visio isaca -p twitter  
facebook instagram badoo
```

Búsquedas utilizando un fichero de nicks (uno por línea):

```
> python usufy.py --list ./test.txt -p all
```

Búsquedas en plataformas de tipo 'social' usando 20 hilos:

```
> python usufy.py -n i3visio -t social -T 20
```

# Creación de *wrappers*

```
class <DEMO>(Platform):
    """
        A <Platform> object for Demo.
    """
    def __init__(self):
        """
            Constructor...
        """
        self.platformName = "<DEMO>"
        self.tags = ["demo", "test"]

        #####
        # Defining valid modes #
        #####
        self.isValidMode = {}
        self.isValidMode["phonefy"] = False
        self.isValidMode["usufy"] = False
        self.isValidMode["searchfy"] = False

        #####
        # Search URL for the different modes #
        #####
        # Strings with the URL for each and every mode
        self.url = {}
        self.url["phonefy"] = "http://anyurl.com/phone/" + "<phonefy>"
        self.url["usufy"] = "http://anyurl.com/user/" + "<usufy>"
        self.url["searchfy"] = "http://anyurl.com/search/" + "<searchfy>"

        #####
        # Whether the user needs credentials #
        #####
        self.needsCredentials = {}
        self.needsCredentials["phonefy"] = False
        self.needsCredentials["usufy"] = False
        self.needsCredentials["searchfy"] = False

        #####
        # Valid queries #
        #####
        # Strings that will imply that the query number is not appearing
        self.validQuery = {}
        # The regular expression '.*' will match any query.
        self.validQuery["phonefy"] = re.compile(".*")
        self.validQuery["usufy"] = re.compile(".*")
        self.validQuery["searchfy"] = re.compile(".*")

        #####
        # Not_found clues #
        #####
```

# Creación de *wrappers*

**Paso 1:** Actualización del nombre de la plataforma.

**Paso 2:** Poner a **True** el tipo de búsquedas que se van a realizar.

**Paso 3:** Introducir la URL base en la que se encontrarán los perfiles con el *tag* **<usufy>**.

**Paso 4:** Indicar si es necesario utilizar credenciales para acceder a dicha dirección URL.

```
class <DEMO>(Platform):  
    """  
        A <Platform> object for Demo.  
    """  
    def __init__(self):  
        """  
            Constructor...  
        """  
        self.platformName = "<DEMO>"  
        self.tags = ["demo", "test"]  
  
        #####  
        # Defining valid modes #  
        #####  
        self.isValidMode = {}  
        self.isValidMode["phonefy"] = False  
        self.isValidMode["usufy"] = False  
        self.isValidMode["searchfy"] = False  
  
        #####  
        # Search URL for the different modes #  
        #####  
        # Strings with the URL for each and every mode  
        self.url = {}  
        self.url["phonefy"] = "http://anyurl.com/phone/" + "<phonefy>"  
        self.url["usufy"] = "http://anyurl.com/user/" + "<usufy>"  
        self.url["searchfy"] = "http://anyurl.com/search/" + "<searchfy>"  
  
        #####  
        # Whether the user needs credentials #  
        #####  
        self.needsCredentials = {}  
        self.needsCredentials["phonefy"] = False  
        self.needsCredentials["usufy"] = False  
        self.needsCredentials["searchfy"] = False  
  
        #####  
        # Valid queries #  
        #####  
        # Strings that will imply that the query number is not appearing  
        self.validQuery = {}  
        # The regular expression '.*' will match any query.  
        self.validQuery["phonefy"] = re.compile(".*")  
        self.validQuery["usufy"] = re.compile(".*")  
        self.validQuery["searchfy"] = re.compile(".*")  
  
        #####  
        # Not found clues #  
        #####
```

# Creación de *wrappers*

```
Abrir ▾ [icon] demo.py.sample /locally-shared/Dropbox/_herramientas/dev...ols/osr... Guardar [icon] - □ ×

#self.needsCredentials[ "usufy" ] = False
#self.needsCredentials[ "searchfy" ] = False

#####
# Valid queries #
#####
# Strings that will imply that the query number is not appearing
self.validQuery = {}
# The regular expression '.' will match any query.
#self.validQuery[ "phonefy" ] = re.compile( "." )
#self.validQuery[ "usufy" ] = re.compile( "." )
#self.validQuery[ "searchfy" ] = re.compile( "." )

#####
# Not_found clues #
#####
# Strings that will imply that the query number is not appearing
self.notFoundText = {}
#self.notFoundText[ "phonefy" ] = []
#self.notFoundText[ "usufy" ] = []
#self.notFoundText[ "searchfy" ] = []

#####
# Fields to be searched #
#####
self.fieldsRegExp = {}

# Definition of regular expressions to be searched in phonefy mode
#self.fieldsRegExp[ "phonefy" ] = {}
# Example of fields:
#self.fieldsRegExp[ "phonefy" ][ "i3visio.location" ] = ""

# Definition of regular expressions to be searched in usufy mode
#self.fieldsRegExp[ "usufy" ] = {}
# Example of fields:
#self.fieldsRegExp[ "usufy" ][ "i3visio.location" ] = ""

# Definition of regular expressions to be searched in searchfy mode
#self.fieldsRegExp[ "searchfy" ] = {}
# These two fields are REQUIRED to grab the results
#self.searchfyDelimiterStart = "<div class='page-listing col-sm-12'>"
#self.searchfyDelimiterEnd = "<div class='row'>"
# These rest of fields to extract
#self.fieldsRegExp[ "searchfy" ][ "i3visio.location" ] = ""

#####
# Fields found #
#####
# This attribute will be feeded when running the program.
self.foundFields = {}
```

Python ▾ Anchura del tabulador: 4 ▾ Ln 78, Col 28 ▾ INS

# Creación de *wrappers*

**Paso 5:** Indicar la expresión regular que deberán satisfacer los nombres de usuario de dicha plataforma para evitar los *nicks* que contengan caracteres no permitidos.

**Paso 6:** Introducir el mensaje de error que figura en el recurso descargado cuando se produce el error.

**Paso 7 [opcional]:** Introducir las expresiones regulares para extraer campos concretos de las plataformas.

```
Abrir ▾ [icon] demo.py.sample /locally-shared/Dropbox/_herramientas/dev...ols/osr... Guardar [icon] - □ ×

#self.needsCredentials[ 'usufy' ] = False
#self.needsCredentials[ "searchfy" ] = False

#####
# Valid queries #
#####
# Strings that will imply that the query number is not appearing
self.validQuery = {}
# The regular expression '.' will match any query.
#self.validQuery[ "phonefy" ] = re.compile('.*')
#self.validQuery[ "usufy" ] = re.compile('.*') ← 5
#self.validQuery[ "searchfy" ] = re.compile('.*')

#####
# Not_found clues #
#####
# Strings that will imply that the query number is not appearing
self.notFoundText = {}
#self.notFoundText[ "phonefy" ] = []
#self.notFoundText[ "usufy" ] = [] ← 6
#self.notFoundText[ "searchfy" ] = []

#####
# Fields to be searched #
#####
self.fieldsRegExp = {}

# Definition of regular expressions to be searched in phonefy mode
#self.fieldsRegExp[ "phonefy" ] = {}
# Example of fields:
#self.fieldsRegExp[ "phonefy" ][ "i3visio.location" ] = ""

# Definition of regular expressions to be searched in usufy mode
#self.fieldsRegExp[ "usufy" ] = {}
# Example of fields:
#self.fieldsRegExp[ "usufy" ][ "i3visio.location" ] = "" ← 7

# Definition of regular expressions to be searched in searchfy mode
#self.fieldsRegExp[ "searchfy" ] = {}
# These two fields are REQUIRED to grab the results
#self.searchfyDelimiterStart = "<div class='page-listing col-sm-12'>"
#self.searchfyDelimiterEnd = "<div class='row'>"
# These rest of fields to extract
#self.fieldsRegExp[ "searchfy" ][ "i3visio.location" ] = ""

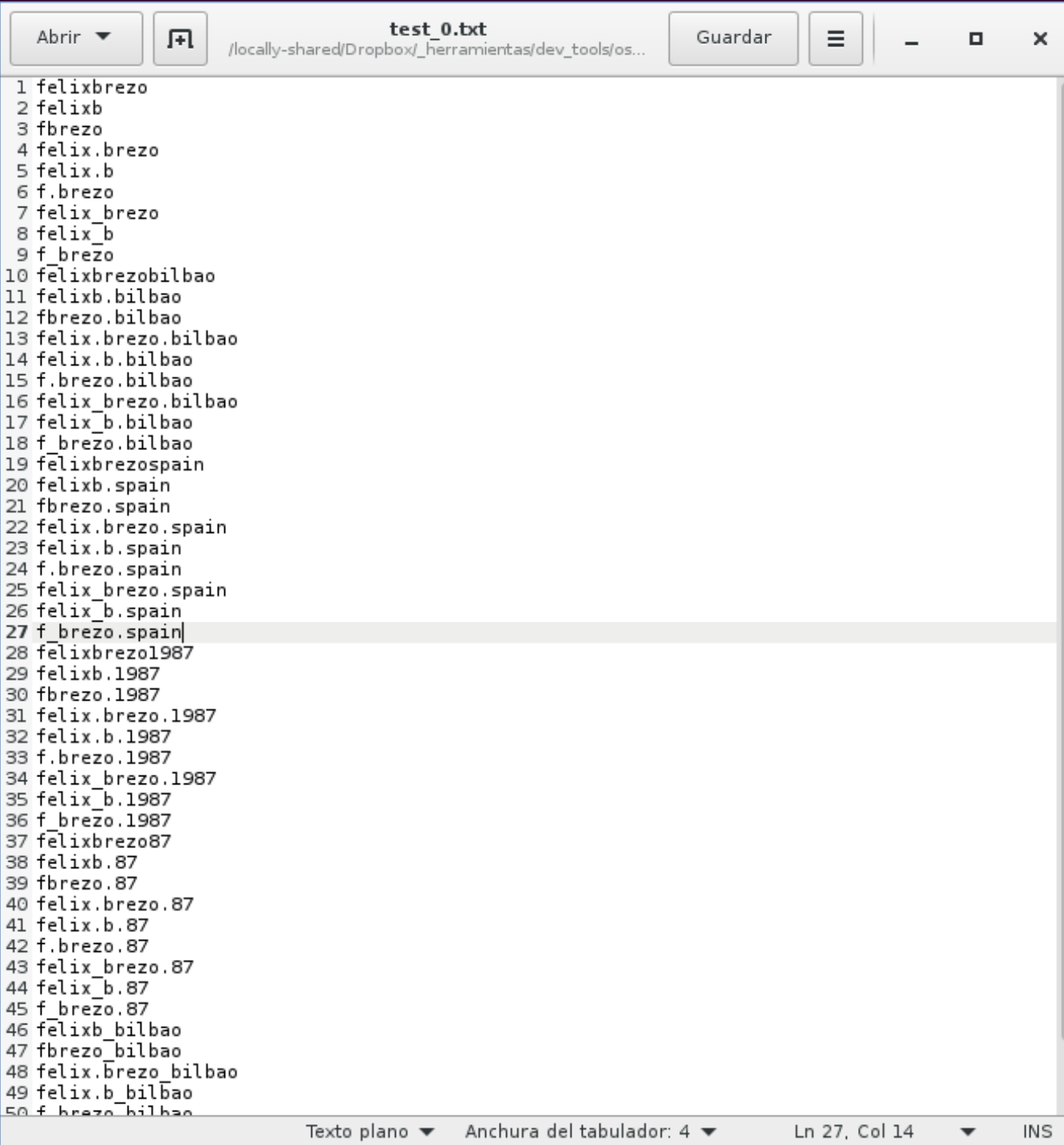
#####
# Fields found #
#####
# This attribute will be feeded when running the program.
self.foundFields = {}

Python ▾ Anchura del tabulador: 4 ▾ Ln 78, Col 28 ▾ INS
```

# Generación de alias candidatos

En base a preguntas básicas se genera una primera lista de nombres de usuario candidatos.

Esta lista puede ser utilizada como entrada para otras aplicaciones de verificación.



The screenshot shows a text editor window titled 'test\_0.txt' with a file path of '/locally-shared/Dropbox/\_herramientas/dev\_tools/os...'. The editor contains a list of 50 candidate usernames, each on a new line and numbered from 1 to 50. The list includes variations of 'felixbrezo', 'felixb', 'fbrezo', 'felix.brezo', 'felix.b', 'f.brezo', 'felix\_brezo', 'felix\_b', 'f\_brezo', 'felixbrezobilbao', 'felixb.bilbao', 'fbrezo.bilbao', 'felix.brezo.bilbao', 'felix.b.bilbao', 'f.brezo.bilbao', 'felix\_brezo.bilbao', 'felix\_b.bilbao', 'f\_brezo.bilbao', 'felixbrezospain', 'felixb.spain', 'fbrezo.spain', 'felix.brezo.spain', 'felix.b.spain', 'f.brezo.spain', 'felix\_brezo.spain', 'felix\_b.spain', 'f\_brezo.spain', 'felixbrezo1987', 'felixb.1987', 'fbrezo.1987', 'felix.brezo.1987', 'felix.b.1987', 'f.brezo.1987', 'felix\_brezo.1987', 'felix\_b.1987', 'f\_brezo.1987', 'felixbrezo87', 'felixb.87', 'fbrezo.87', 'felix.brezo.87', 'felix.b.87', 'f.brezo.87', 'felix\_brezo.87', 'felix\_b.87', 'f\_brezo.87', 'felixb\_bilbao', 'fbrezo\_bilbao', 'felix.brezo\_bilbao', 'felix.b\_bilbao', and 'f\_brezo\_bilbao'. The status bar at the bottom indicates 'Texto plano', 'Anchura del tabulador: 4', 'Ln 27, Col 14', and 'INS'.

```
1 felixbrezo
2 felixb
3 fbrezo
4 felix.brezo
5 felix.b
6 f.brezo
7 felix_brezo
8 felix_b
9 f_brezo
10 felixbrezobilbao
11 felixb.bilbao
12 fbrezo.bilbao
13 felix.brezo.bilbao
14 felix.b.bilbao
15 f.brezo.bilbao
16 felix_brezo.bilbao
17 felix_b.bilbao
18 f_brezo.bilbao
19 felixbrezospain
20 felixb.spain
21 fbrezo.spain
22 felix.brezo.spain
23 felix.b.spain
24 f.brezo.spain
25 felix_brezo.spain
26 felix_b.spain
27 f_brezo.spain
28 felixbrezo1987
29 felixb.1987
30 fbrezo.1987
31 felix.brezo.1987
32 felix.b.1987
33 f.brezo.1987
34 felix_brezo.1987
35 felix_b.1987
36 f_brezo.1987
37 felixbrezo87
38 felixb.87
39 fbrezo.87
40 felix.brezo.87
41 felix.b.87
42 f.brezo.87
43 felix_brezo.87
44 felix_b.87
45 f_brezo.87
46 felixb_bilbao
47 fbrezo_bilbao
48 felix.brezo_bilbao
49 felix.b_bilbao
50 f_brezo_bilbao
```

## Identificación de entidades adicionales

Las **expresiones regulares** pueden ayudar a la identificación de información. En OSRFramework se incluye **entify.py**, *script* que facilita la recuperación de entidades presentes en recursos web utilizando expresiones regulares:

- Direcciones de correo electrónico: foo@bar.com, foo[at]bar[dot]com, foo[arroba]bar[punto]com, etc.
- Direcciones URL
- Direcciones IPv4
- DNI
- Cuentas de criptodivisas: Bitcoin, Litecoin, Peercoin, Dogecoin.
- Hashes: MD5, SHA1, SHA256.



## Identificando entidades usando **entify.py**

Búsquedas en un sitio web de todas las entidades de tipo email:

```
> python entify.py -r i3visio.email -w http://i3visio.com
```

Búsquedas en un sitio web de las entidades que matchean con todas las expresiones regulares cargadas:

```
> python entify.py -r all -w http://i3visio.com
```

Búsquedas en un sitio web de exp. facilitadas por el usuario:

```
> python entify.py -R "([0-9]{8})" -w http://i3visio.com
```

Búsquedas en los ficheros de la carpeta **test** de todas las expresiones regulares cargadas:

```
> python entify.py -r all -i ./test/
```



## Otras utilidades de OSRFramework

Se han desarrollado otras utilidades que facilitan las tareas de investigación:

**mailfy.py** → aplicación para verificar la existencia o no de un correo electrónico de Gmail o Hushmail.

**phonefy.py** → aplicación para identificar la vinculación de un número de teléfono con llamadas de *spam* telefónico.

**searchfy.py** → aplicación para realizar búsquedas en distintas plataformas, tanto de la web de superficie como de redes anónimas.

**enumeration.py** → prueba de concepto para plantear la recolección de usuarios de diferentes plataformas por el método de *user enumeration*.

## Obtención de información de API, bases de datos y servicios de terceros

Adicionalmente, se han incluido otras utilidades de terceros que facilitan las tareas de investigación con respecto a determinadas entidades:



**BLOCKCHAIN**

# 4

Visualización a través  
de transformadas



## Configuración de las transformadas

Además del modo consola, se puede usar una interfaz gráfica para visualziar las relaciones. En este caso, **Maltego**.

Para ello es necesario añadir un paso adicional:

**Paso 1:** Descargar OSRFramework

**Paso 2:** Descomprimir OSRFramework

**Paso 3:** Instalar los módulos

Y...

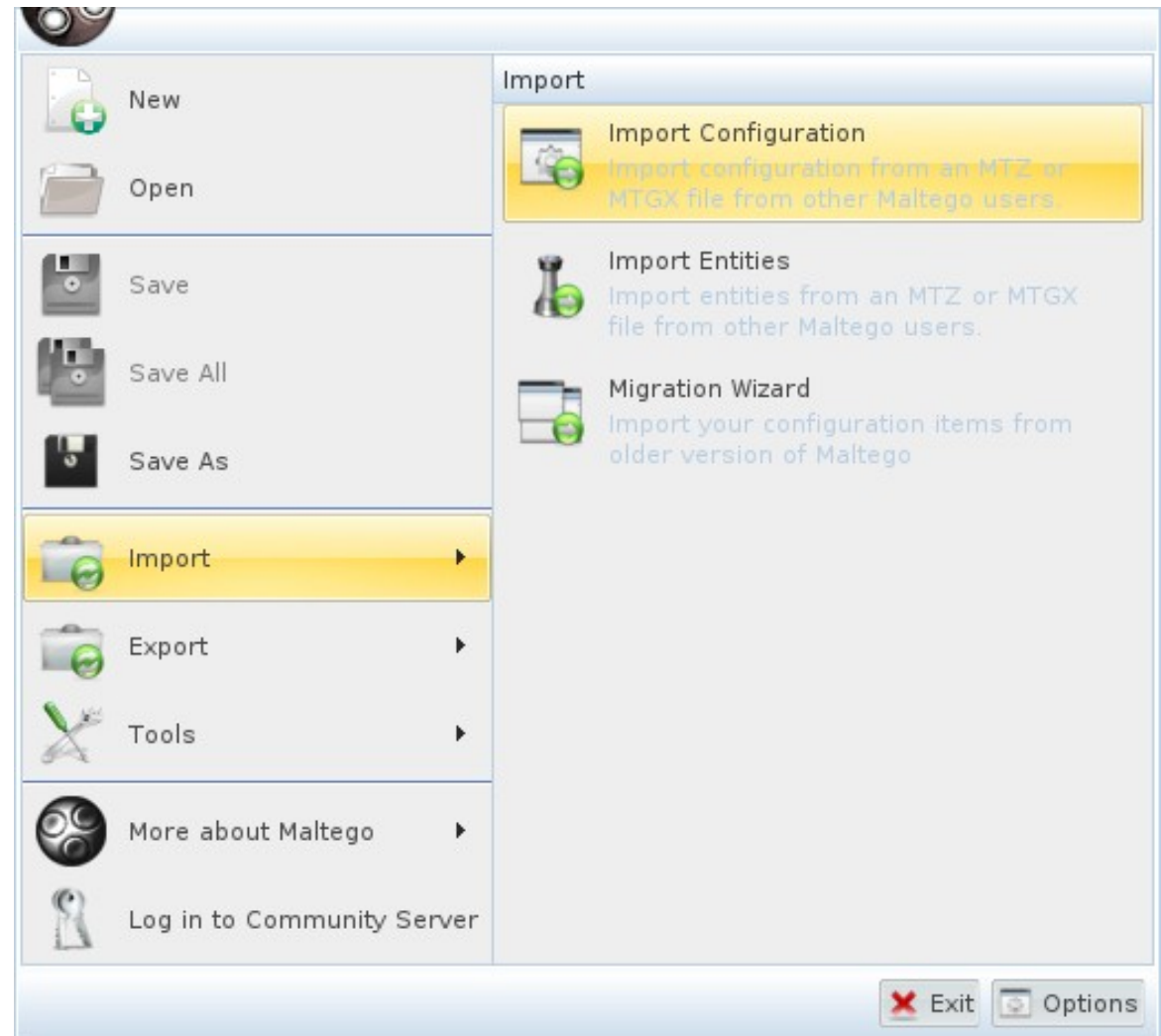
**Paso 4 [Nuevo]:** Generar los ficheros de configuración desde la carpeta de instalación:

```
> python configure_maltego.py
```

## Configuración de las transformadas

Desde el cliente de **Maltego** será necesario ahora importar la nueva configuración generada por el *script* de configuración.

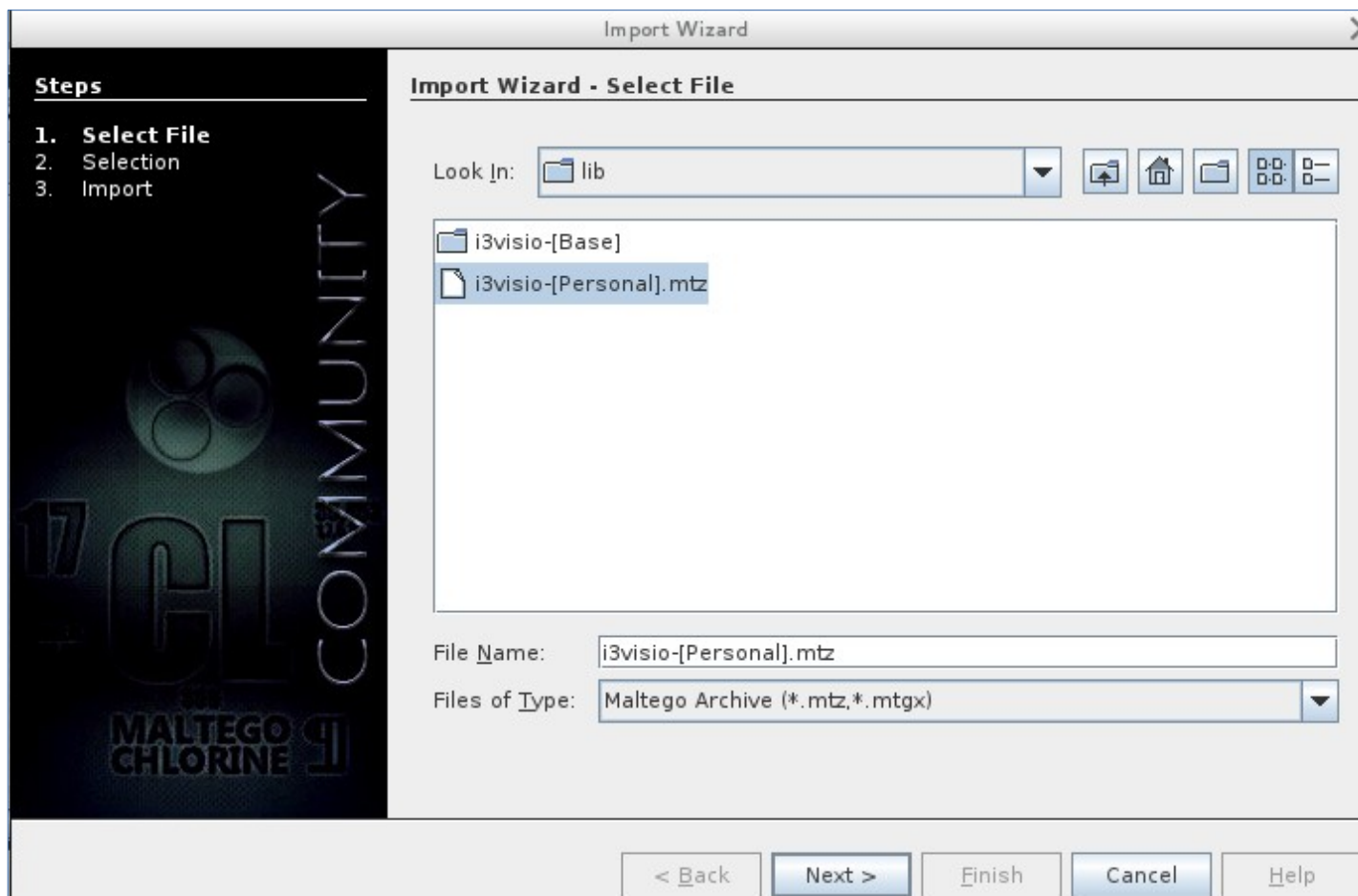
Este *script* genera las entidades y transformadas enlazadas con la ruta actual de instalación de **OSRFramework**.



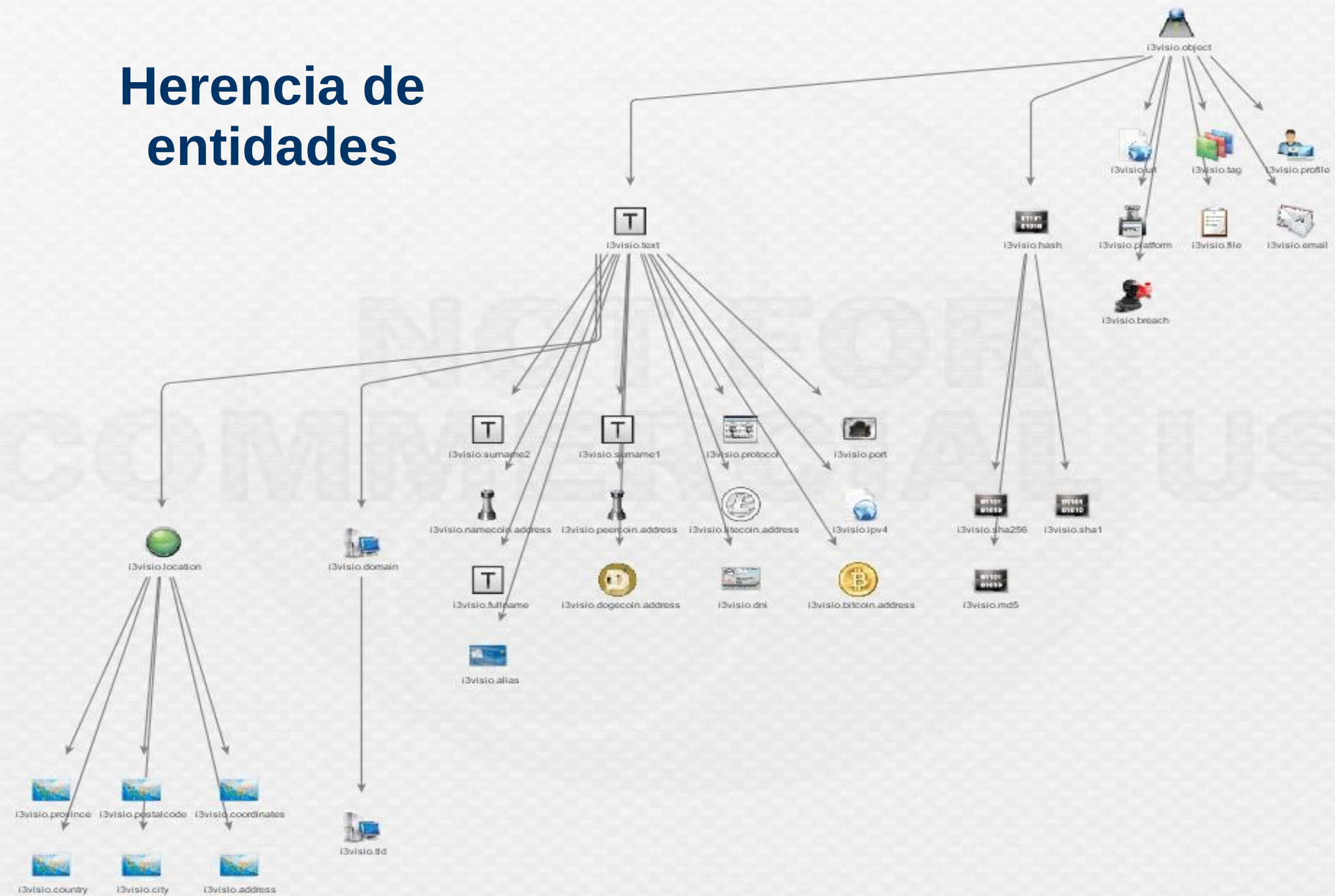
# Configuración de las transformadas

Navegamos hasta la siguiente carpeta para importar el archivo **.mtz**:

**<BASE\_FOLDER>/osrframework/transforms/lib**

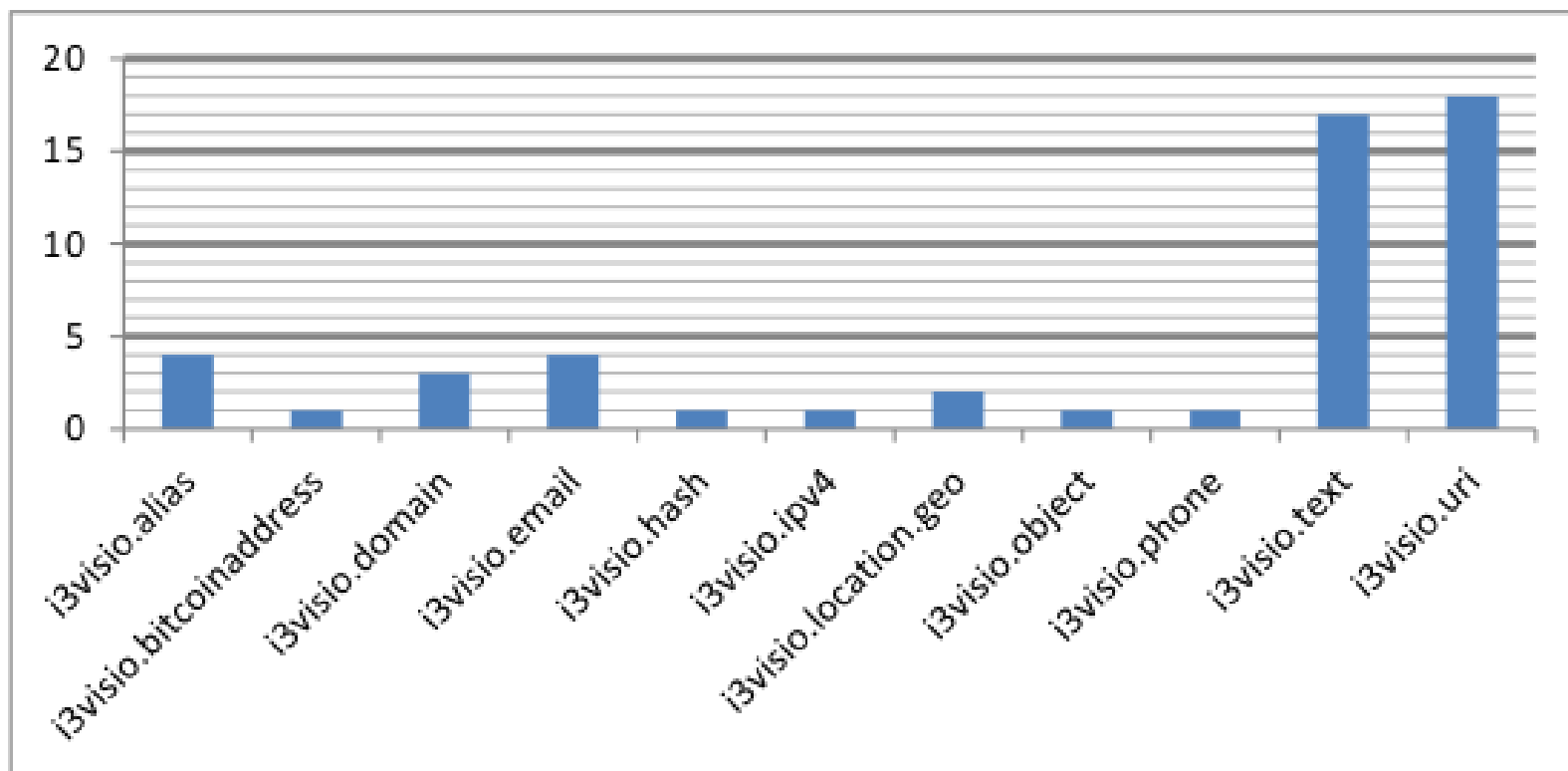


# Herencia de entidades





## Transformadas incluidas en función de la entidad de origen



**Figura 3.** Desglose de las transformadas generadas en función de la entidad de origen.



5

Caso de estudio



6

Líneas de  
trabajo futuras



## Líneas de trabajo futuro

Integración de **nuevas plataformas** para la identificación de usuarios

Extracción de campos de todas las plataformas identificadas con **expresiones regulares**

**Normalización** y equiparación de los parámetros

Ampliación de las capacidades para la **realización de búsquedas** dentro de las plataformas empleando searchfy.py

Integración de **API de terceros** que permitan el enriquecimiento de la información identificada con bases de datos adicionales adecuadas al modelo de datos propuesto por el *framework*

Adición de **nuevas transformadas** y formas de representación de la información



# Conclusiones





La **diversidad** de plataformas existente dificulta las labores de investigación en la red

El seguimiento de la actividad se puede tornar **caótico** para el investigador si no dispone de procedimientos

Además, no siempre se cuenta con API y herramientas apropiadas para satisfacer **necesidades concretas**





OSRFramework se traduce  
en una **herramienta de  
investigación** para dar  
respuesta a necesidades  
concretas de los analistas



En el caso práctico se ha puesto de manifiesto que este *framework* sirve para **agilizar el proceso** de identificación de ciberidentidades a partir de la información recogida en el proceso de investigación previo





# #RetoISACA2015

OSRFramework, un *framework*  
libre para la investigación de  
usuarios en fuentes abiertas

Yaiza Rubio (yaiza\_rv@hotmail.com)

Félix Brezo (felixbrezo@gmail.com)

