Sploitego
Maltego’s (Local) Partner in Crime

Nadeem Douba | www.cygnos.com
Introduction

How you doin’
About Me

Nadeem Douba

- Work at Cygnos ([http://www.cygnos.com](http://www.cygnos.com)) in Ottawa, ON, Canada
- Certs: GWAPT, GPEN
- Worked in the InfoSec field for 10+ years.
- Love (European) football and hacking stuff…

- Been a Maltego fan–boy since the beginning…
- Helped port/appify Maltego for Mac OS X 😊
Presentation Overview

- What is Sploitego?
- Maltego – Briefly Explained
- Dive Into Development
  - Before Sploitego
  - After Sploitego
- Demos
- Conclusion
- Questions
What is Sploitego?

- Local Transform Development Framework for Maltego written in Python
- Provides:
  - Rapid transform development
  - Easy transform installation, management, and maintenance
  - Complementary scripts and modules for data mining and debugging
  - A whole bunch of cool transforms 😊
But First...

- A little background on Maltego...
Background

Maltego Overview
What is Maltego?

- Open Source Intelligence (OSInt) and forensics information mining/gathering and graphing tool
- Developed by Paterva and PinkMatter
What is Maltego? – Cont.

- Information is represented on the graph as **Entities**
- For example, an **Entity** could be:
  - Email Address
  - Image
  - Phone Number
  - Etc.
- Each **Entity** has a value and optionally some fields.
What is a Transform?

- **Transforms** reveal relationships between entities (or information)
- Logic that mines and returns information (or Entities) using another piece of information (or Entity) as input
  - \[ T(E_0) \rightarrow \{ E_1, E_2, \ldots, E_n \} \]
Maltego Demo

Just for Clarity
Two types of transforms:

- **Remote**: runs on a remote Paterva or third-party Transform Server.
- **Local**: runs on the user’s local machine.
  - This is where Sploitego comes in...
Remote Transforms – Pros & Cons

**Pros**

- Paterva’s Transforms
  - Awesome!!!
- Centralized Transform Management & Maintenance
  - Implementation details hidden from the user (protects your IP)
- Minimal Client–Side Processing Overhead

**Cons**

- Limited Data Visibility
  - i.e. Server can only query accessible data.
- Breach of Privacy
  - OSInt target/subject disclosed to a third–party.
- Limited Client–Side Control:
  - Transforms might not be evil enough 😊
Local Transforms – Pros and Cons

**Pros**

- Full Client-side Control
  - No limits as to how 1337 or evil your transforms can be 😊
- Privacy
  - OSInt subject may not be disclosed to third-party
- Great Data Visibility
  - “The world is one’s oyster”
- Extensible
  - Maltego can be used for other types of data visualization 😊

**Cons**

- Processing Overhead
  - Client’s machine responsible for running transforms
- Development
  - It’s in your hands (or somebody else’s... just delegate ;)
- IP Disclosure
  - Implementation details no longer hidden from users.
- Difficult to Maintain
Local Transform Development

The Nitty Gritty
How do Local Transforms Work?

- Maltego executes a local script or executable
- Input passed via command line arguments:
  
  ```
  $ ./mytransform.sh <value>
  <fieldname_1>=<fieldvalue_1>...<fn_n>=<fv_n>
  ```

- Transform results returned via standard output in Maltego XML message format

- Debugging messages returned via standard error
Example – Transform Call

$ ./t.pl aspmx.l.google.com
mxrecord.priority=0

• **Note:** the bolded property ("MX Record"), below, is the entity value (or Display Value)
Example – Transform Message

```xml
<MaltegoMessage>
  <MaltegoTransformResponseMessage>
    <Entities>
      <Entity Type="maltego.IPv4Address">
        <Value>0.0.0.0</Value>
        <Weight>1</Weight>
        <AdditionalFields>
          <Field DisplayName="Internal"
                 MatchingRule="strict"
                 Name="ipaddress.internal">true</Field>
          <Field DisplayName="Hardware Address"
                 MatchingRule="strict"
                 Name="ethernet.hwaddr">00:00:00:00:00:00</Field>
        </AdditionalFields>
      </Entity>
    </Entities>
  </MaltegoTransformResponseMessage>
</MaltegoMessage>
```
Writing a Local Transform

Without Sploitego
Local Transform Development Checklist

- Learn Maltego Local Transform Specification
  - XML Messaging
  - Debugging
  - Etc.

- Develop Transform
  - Input Parsing Logic
  - Data Mining Logic
  - XML Serialization Logic
  - Debugging Facilities

- Install Transform

- Configure & Maintain Transform

- Define Entity in Maltego (Optional)
Example – Hello World Transform

- 47 lines of code for a simple transform
  - Not bad…
  - But not great either
- XML is hard-coded
  - Not reusable
  - Debugging nightmare!
- Imagine returning 100+ entities with fields 😊
Installing Transforms

- Currently Manual Process
  - Two-step Wizard per Transform
- Tedious & Prone to User Error
  - More Transforms = More Configuration = Less Time Playing
Grouping Transforms

- Have to manually create a *Transform Set*
- Another dialog box somewhere 😞
- When does the fun begin?
What is Sploitego?

- Local Transform Development Framework for Maltego written in *Python*
- Provides:
  - Rapid transform development
  - Easy transform installation, management, and maintenance
  - Complementary scripts and modules for data mining and debugging
  - A whole bunch of cool transforms 😊
- How does it bring back the fun?
Remember our Checklist?

- Learn Maltego Local Transform Specification
  - XML Messaging
  - Debugging
  - Etc.
- Develop Transform
  - Input Parsing Logic
    - Data Mining Logic ← This is all you have to take care of! – Wawa–wiwa!
  - XML Serialization Logic
  - Debugging Facilities
- Install Transform
- Configure & Maintain Transform
  - Define Entity in Maltego (Optional) ← And possibly this...
Sploitego Transforms – Packaging

- Sploitego transforms are simply **Python Modules** within **Python Packages**
- Follows traditional Python package directory structure:
  - `.setup.py` (Python installation script – distutils/setuptools)
  - `/foobar` (Package directory)
  - `/foobar/__init__.py` (Module/package init script)
  - `/foobar/helloworld.py` (Transform module)
Hello World (Revised) Transform

```python
from sploitego.maltego.message import Person, Phrase
from sploitego.maltego.utils import import debug, progress
from sploitego.framework import import superuser, configure

@superuser
@configure(
    label='To Phrase [Hello World]',
    description='Returns a phrase entity with the phrase "Hello World!"
    uuids=[ 'sploitego.v2.PersonToPhrase_HelloWorld' ],
    inputs=[ ( 'Useless', Person ) ],
    debug=True
)

def dotransform(request, response):
    progress(50)
    debug('This was pointless!')
    progress(100)
    return response + Phrase('Hello %s' % request.value)

def onterminate():
```

Additional Steps

- `foobar/__init__.py` must contain `__all__` variable with transform modules specified.

```python
#!/usr/bin/env python
__all__ = ['helloworld']
```
Dissecting the Transform
The `dotransform` function is the entry point

Accepts two parameters: `request`, and `response`

The `request` object has the following properties:
- `value`: the Entity display value (string)
- `fields`: the Entity fields (dictionary)
- `params`: extra parameters that can be parsed by `optparse`
The `response` object is where we populate our results

`dotransform` must return the `response` object

`response` object uses mathematical operators to add and remove `Entity` and `UIMessage` objects
- E.g. `response + Phrase('Hi')` appends a Phrase Entity object to the response object

Finally, `onterminate` function is called if `Maltego` interrupts the transform – it is optional
@superuser instructs the dispatcher to run the transform as the super-user

If a transform is being executed as a non-super-user:
- dispatcher will invoke sudo
- Prompt user for sudo password
- If successful, execute the transform using sudo
- Else, abort execution after three retries
Installation Meta-Data – \texttt{@configure}

- Instructs \texttt{mtginstall} on how to configure transform in Maltego
- **Parameters:**
  - \texttt{label}: display label of transform in Maltego
  - \texttt{description}: A brief description
  - \texttt{uuids}: list of universally unique identifiers (or transform descriptor file names)
  - \texttt{inputs}: list of tuples containing \textit{Transform Set} name and \textit{Input Entity} type
  - \texttt{debug}: whether or not debug window should appear in Maltego on transform execution
Installation Meta-Data – \@configure – cont.

For example:
- Transform will appear as \texttt{To Phrase [Hello World]} in Maltego GUI
- Will belong to the \texttt{Useless} Transform Set
- Can only be applied to \texttt{Person} type Entities
- Have a unique ID of \texttt{sploitego.v2.PersonToPhrase_HelloWorld}
- A debug window will appear on transform execution
Notice how `uuids` and `inputs` are lists

`mtginstall` supports one-to-many relationship between transforms and input entity types

- For example, *Hello World Transform* could be applied to *Phrase* entities as well
- Just add another `uuid` and `inputs` entry (matching order)

```python
@configure(
    label='To Phrase [Hello World]',
    description='Returns a phrase entity with the phrase "Hello World!"',
    uuids= ['sploitego.v2.PersonToPhrase_HelloWorld',
            'sploitego.v2.PhraseToPhrase_HelloWorld'],
    inputs= [( 'Useless', Person ),
              ( 'Useless', Phrase ) ],
    debug=True
)
```
Hello World (Revised) – The Stats

- **24 Lines of Code in Total!**
  - Approximately 50% less code!
  - Only **SIX (6)** lines were “actual” code!
  - The rest were annotations, function signatures, and imports
- Not a single print line in sight!
- No hard-coded XML!
- What about installation?
Managing Transform Packages

Install, Uninstall, Etc.
To install a Sploitego transform:

- **First**, install Python package containing transforms
  - distutils or setuptools are great for that!
- **Alternatively**, place Python module in Maltego’s working directory
- **Second**, run `mtginstall`
Installing Transforms (Revised) – Cont.

- **Input Parameters:**
  - *Hello World Transform* is in *foobar* package
  - Maltego’s settings are stored in `~/Library/Application\ Support/maltego/v3.1.1/` (on Mac OS X)
  - Your transform working directory is `~/

- **To Install Transform Package, Run:**
  ```
  $ mtginstall --package foobar --maltego-prefix
  ~/Library/Application\ Support/maltego/v3.1.1/
  --working-dir ~/
  ```
Transform Installer – mtginstall

1. mtginstall first imports __init__.py in foobar package
2. Iterates the __all__ special variable to get list of modules in package
3. Loads each module and looks for dotransform function annotated with @configure
4. Reads installation meta-data and installs transform in Maltego accordingly
   a) If Transform Set doesn’t exist, it will create it.
   b) Detects name collisions between transforms
Uninstalling Transforms

To uninstall a Sploitego transform run

```
mtguninstall:
$ mtguninstall --package foobar --maltego-prefix ~/Library/Application\ Support/maltego/v3.1.1/
```

`mtguninstall` will remove the transform package (Transform Sets and Transforms) from Maltego’s GUI but not from Python site-package directory.
Demos

The Fun Stuff
Metasploit Integration

Demo
Nmap/Amap Integration

Demo
Scapy Integration

Demo
Extra Utilities

The Goodies
Debugging, Testing, Etc.

- **mtgdebug** script prints results in readable format
- **mtgsh** shell version of **mtgdebug** – still a work in progress

```
bitter:~ ndouba$ mtgdebug sploitego.transforms.whatismyip -
'-- MaltegoTransformResponseMessage:
'-- Entities:
  '-- Entity: {'Type': 'maltego.IPv4Address'}
  '-- Value: 0.0.0.0
  '-- Weight: 1
  '-- AdditionalFields:
    '-- Field: true {'DisplayName': 'Internal', 'Name': 'ipaddress.internal', 'MatchingRule': 'strict'}
    '-- Field: 00:00:00:00:00:00:00 {'DisplayName': 'Hardware Address', 'Name': 'ethernet.hwaddr', 'MatchingRule': 'strict'}
```
Graph Export Conversion Tools

- **mtgx2csv** converts exported Maltego graphs to CSV (comma-separated value) format.
- **csv2sheets** reads the output of mtgx2csv and separates entities of the same type into separate CSVs.
CONCLUSIONS

Last but not Least
Project Roadmap

- Get a website up with some documentation 😊
- Create more transforms for:
  - Social Engineering
  - Forensics
  - Exploitation
  - Scanning and Vulnerability Discovery
  - Third-party Tool Integration
  - Etc.
- Create an online community and transform package index for transform developers similar to PyPI
- Develop a context engine
  - Minimize data duplication on graphs
  - Provide transforms with access to full graph
Looking for Help!

- Sploitego needs your help!
  - Developers
  - Transform Gurus
  - Hackers
  - Documenters
  - Website Designers
  - Chefs who deliver to the Ottawa area 😊
Contact Info

Please feel free to contact me:
- Email: ndouba@gmail.com
- Twitter: @ndouba
- Skype: nadeem.douba
Kudos

- To the Paterva team:
  - Andrew MacPherson (Mohawk)
  - Roelof Temmingh (RT)
- To the Cygnos & RCGT team (w00t!)
- Thank you for attending!
Questions

Anyone?