Hacking Like It's 2013

(with Pythonect & Hackersh)

Itzik Kotler

Hacking Like It's 1999

- Scripts written in different languages - Low Code Reuse
- Change of Logic causes Tool rewrite - Low Code Reuse
- Tools produces different outputs - Agnostic Tools
- Tools takes different parameters and formats - Agnostic Tools
- Multi-threading is a Tool feature - Scaleability Issues
- And etc.

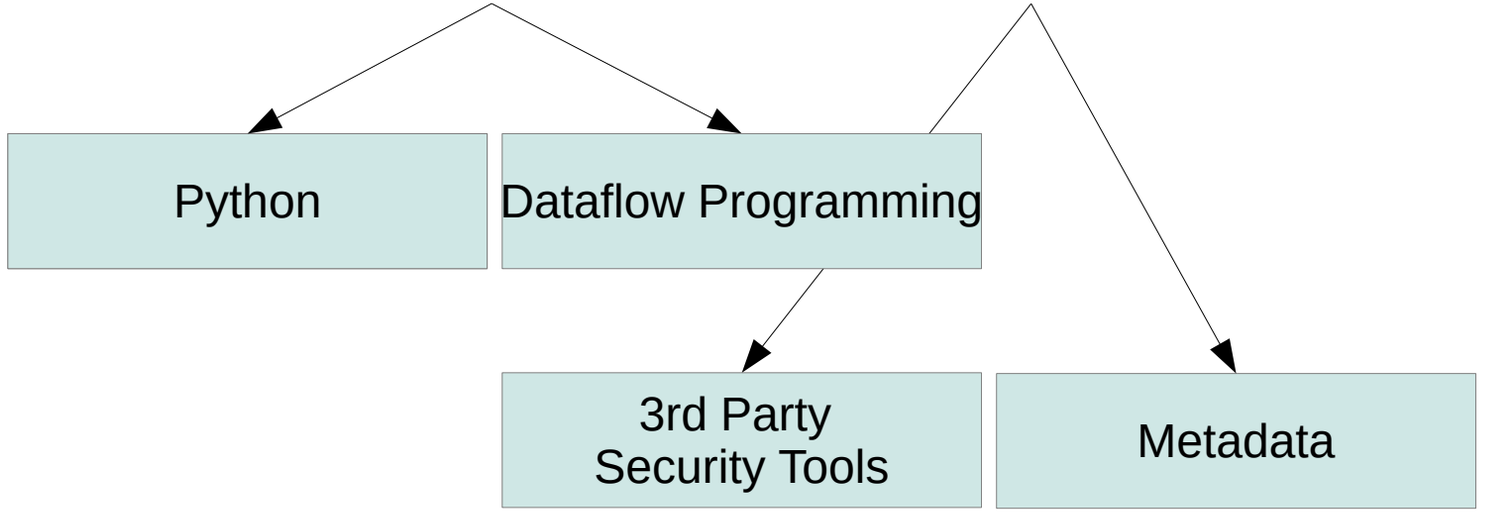
Hacking Like It's 1999 - Summary

- Problems:
  - Agnostic Tools
  - Scalability
  - Code Reuse

Requirements for Hacking Like It's 2013

- Synergy Between Tools
  - Tool can “import“ another tool output/results
  - Tool arguments/parameters are Standardized
- Maximize Code Reuse
  - Special cases do not require re-writing of the Tool
  - Code can be used multiple times in multiple ways
- Scalability (regardless of the Tool used)
  - Multi-threading
  - Multi-processing

`Pythonect` + `Hackersh` = **Hacking Like It's 2013**



Pythonect

- *Pythonect* is a portmanteau of the words Python and Connect
- New, experimental, general-purpose dataflow programming language based on Python
- Current “stable“ version (True to May 19 2013): 0.5.0
- Made available under 'Modified BSD License'
- Influenced by: Unix Shell Scripting, Python, Perl
- Cross-platform (should run on any Python supported platform)
- Website: http://www.pythonect.org/

Installing and Using The Pythonect Interpreter

- Install directly from PyPI using easy_install or pip:
  - `easy_install Pythonect`

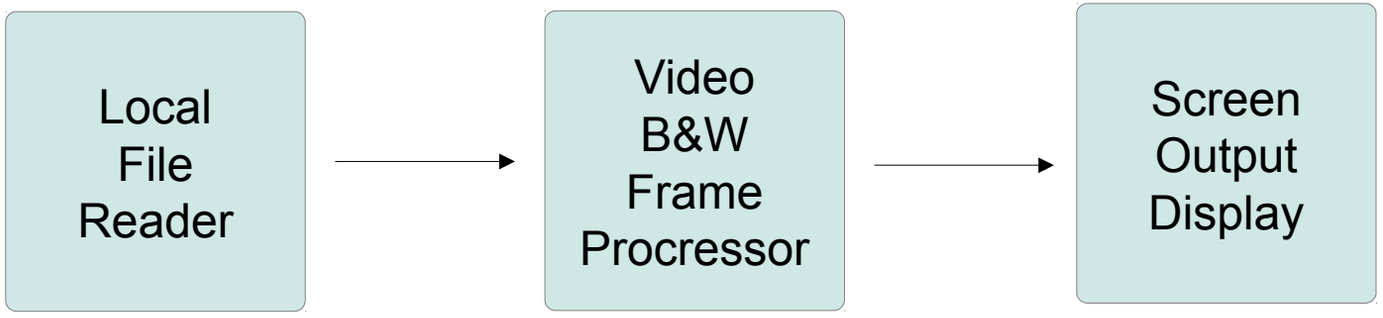
    OR
  - `pip install Pythonect`
- Clone the git repository:
  - `git clone git://github.com/ikotler/pythonect.git`
  - `cd pythonect`
  - `python setup.py install`

Dataflow Programming

Programming paradigm that treats data as something originating from a source, flows through a number of components and arrives at a final destination - most suitable when developing applications that are themselves focused on the "flow" of data.

Dataflow Example

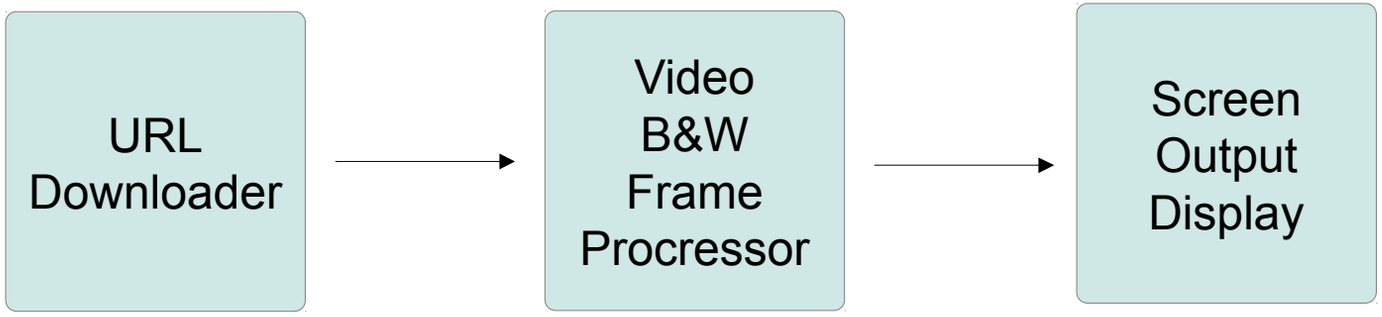
A video signal processor which may start with video input,
modifies it through a number of processing components (i.e. video filters),
and finally outputs it to a video display.



Dataflow Example

Want to change a feed from a local file to a remote file on a website?

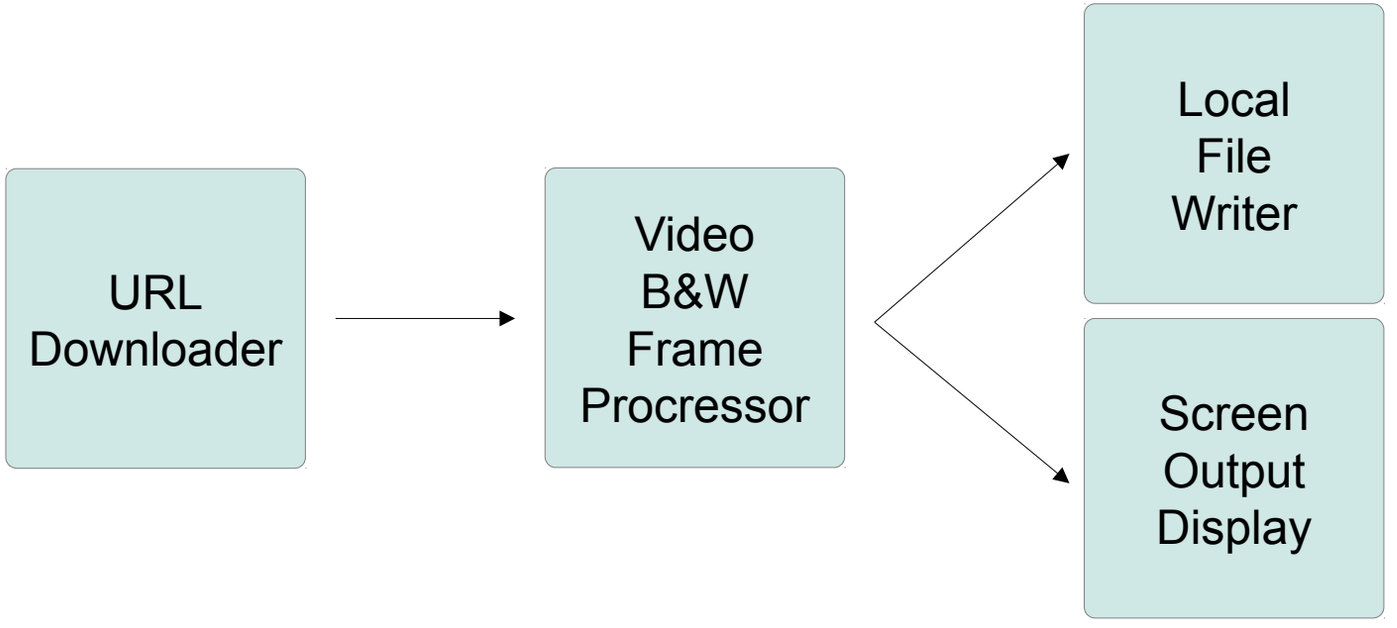
No problem!



Dataflow Example

Want to write the Video B&W Frame Processor output to both a screen and a local file?

No problem!



Dataflow Programming Advantages

- Concurrency and parallelism are natural
- Data flow networks are natural for representing process
- Data flow programs are more extensible than traditional programs

Dataflow Programming Disadvantages

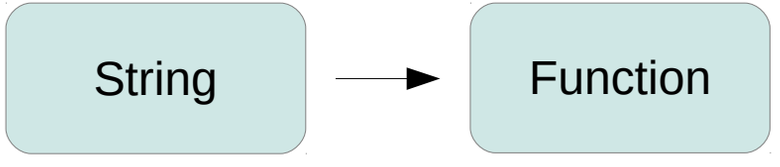
- The mindset of data flow programming is unfamiliar to most programmers
- The intervention of the run-time system can be expensive

Dataflow Programming Languages

- Spreadsheets are essentially dataflow (e.g. Excel)
- VHDL, Verilog and other hardware description languages are essentially dataflow
- XProc
- Max/Msp
- Etc.

<Pythonect Examples>

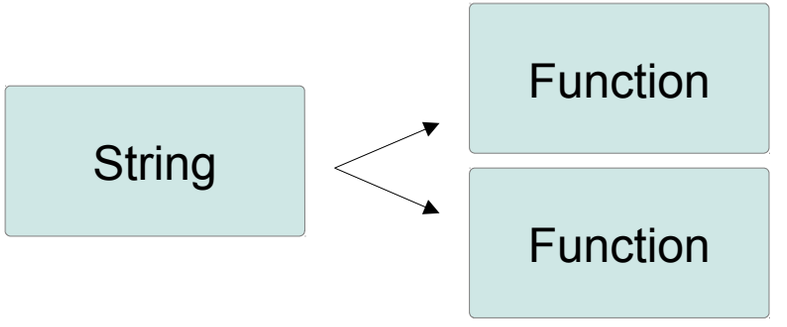
```
'Hello, world' -> print
```



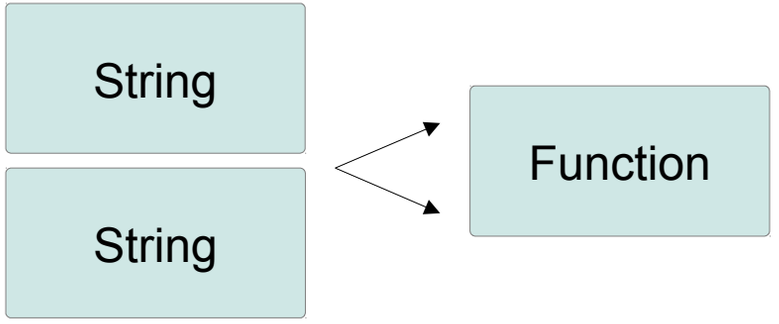
What do we have here?

- `->` is a Pythonect Control Operator, it means async forward.
- There's also `|` (i.e. Pipe) which means sync forward.
- `'Hello, world'` is a literal string
- `print` is a function

```
"Hello, world" -> [print, print]
```



- `["Hello, world", "Hello, world"] -> print`



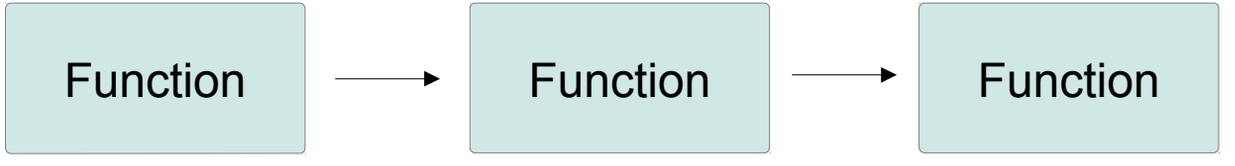
Basic Pythonect Syntax Summary

- `->` is async forward.
- `|` (i.e. Pipe) is sync forward.
- `_` (i.e. Underscore) is current value in flow

<Pythonect Security Scripts/Examples>

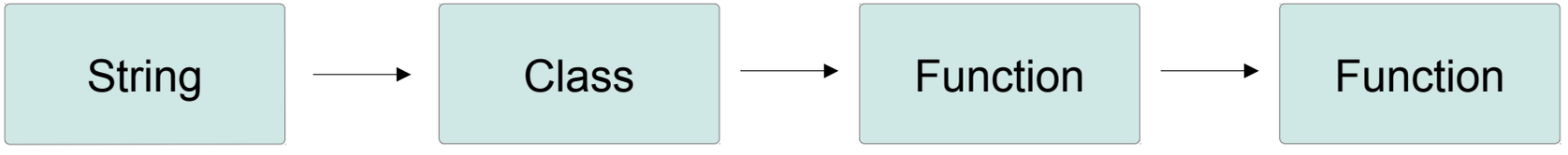
ROT13 Encrypt & Decrypt

```
raw_input() -> _.encode('rot13') -> print
```



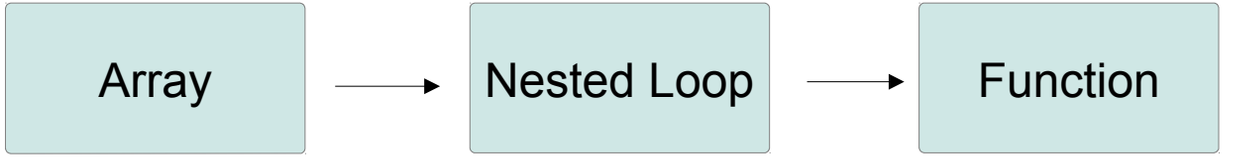
Check if FTP Server Supports Anonymous Login

```
'ftp.gnu.org' \
    -> ftplib.FTP \
    -> _.login() \
    -> print("Allow anonymous")
```



Command line Fuzzer

```
['%s', '%n', 'A', 'a', '0', '!', '$', '%', '*', '+', ',', '-', '.', '/', ':'] \
    | [_ * n for n in [256, 512, 1024, 2048, 4096]] \
        | os.system('/bin/ping ' + _)
```



References / More Examples

- My Blog
  - Scraping LinkedIn Public Profiles for Fun and Profit
  - Fuzzing Like A Boss with Pythonect
  - Automated Static Malware Analysis with Pythonect
- LightBulbOne (Blog)
  - Fuzzy iOS Messages!

-> Moving on! ->

Hackersh

Hackersh

- *Hackersh* is a portmanteau of the words Hacker and Shell
- Shell (command interpreter) written with Pythonect-like syntax, built-in security commands, and out of the box wrappers for various security tools
- Current “stable“ version (True to May 19 2013): 0.2.0
- Made available under GNU General Public License v2 or later
- Influenced by: Unix Shell Scripting and Pythonect
- Cross-platform (should run on any Python supported platform)
- Website: http://www.hackersh.org

A few words on the Development

- Written purely in Python (2.7)
- Hosted on GitHub

Motivation

- ~~Taking over the world~~
- Automating security tasks and reusing code as much as possible

Installing and Using The Hackersh

- Install directly from PyPI using easy_install or pip:
  - `easy_install Hackersh`

    OR
  - `pip install Hackersh`
- Clone the git repository:
  - `git clone git://github.com/ikotler/hackersh.git`
  - `cd hackersh`
  - `python setup.py install`

Implementation

- Component-based software engineering
  - External Components:
    - Nmap
    - W3af
    - Dnsdict6
    - Etc.
  - Internal Components:
    - URL (i.e. Convert String to URL)
    - IPv4_Address (i.e. Convert String to IPv4 Adress)
    - IPv6_Address (i.e. Convert String to IPv6 Adress)
    - Etc.

Component as Application

- Components accepts command line args:
  - `"localhost" -> hostname -> nmap("-P0")`
- They also accept internal flags options as:
  - `"localhost" -> hostname -> nmap("-P0", debug=True)`

Input/Output: Context

- Every Hackersh component (except the Hackersh Root Component) is standardized to accept and return the same data structure – Context.
- Context is a dict (i.e. associative array) that can be piped through different components

Same Context, Different Flow

- `"http://localhost" -> url -> nmap -> ping`
  - Port scan a URL, if *ANY* port is open, ping the URL
- `"http://localhost" -> url -> ping -> nmap`
  - Ping the URL, if pingable, scan for *ANY* open ports

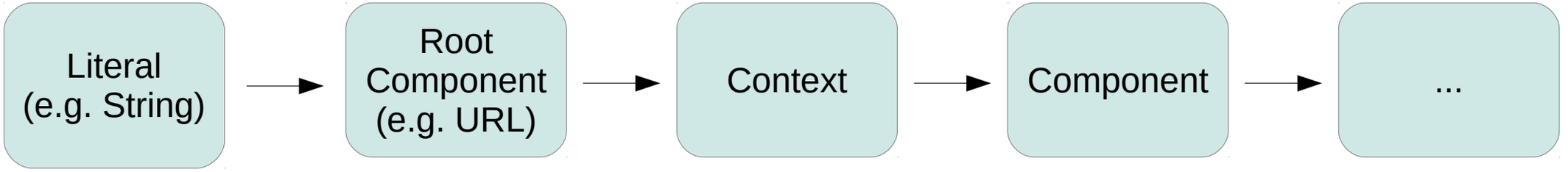
Ask The Context

- Context stores both Data and Metadata
- The Metadata aspect enables potential AI applications to fine-tune their service selection strategy based on service-specific characteristics

Conditional Flow

```
"http://localhost" \
    -> url \
        -> nmap \
            -> [_['PORT'] == '8080' and _['SERVICE'] == 'HTTP'] \
                -> w3af \
                    -> print
```

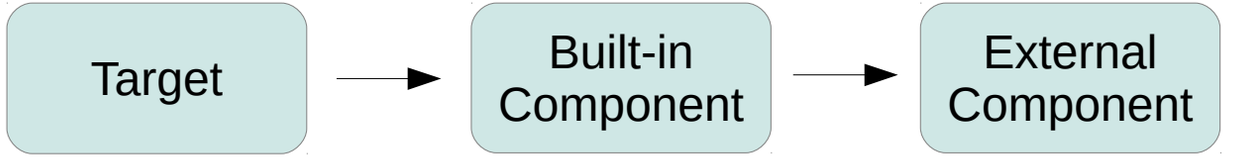
Hackersh High-level Diagram



<Hackersh Web App Pentest Script Step-by-Step>

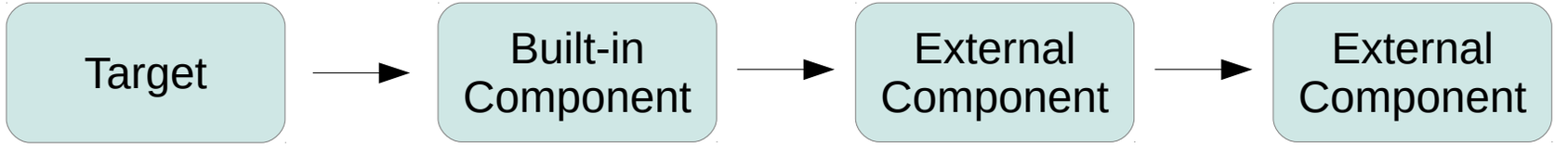
Step #1: Information Gathering / Network Analysis / DNS

```
"hackersh.org" -> domain -> dnsdict(“-4“)
```



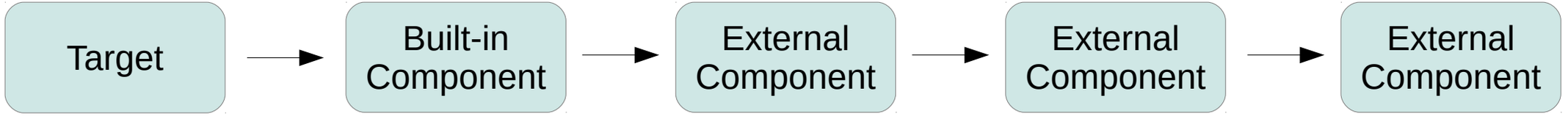
Step #2: Information Gathering / Network Analysis / Service Fingerprinting

```
"hackersh.org" \
    -> domain \
        -> dnsdict(“-4“) \
            -> nmap
```



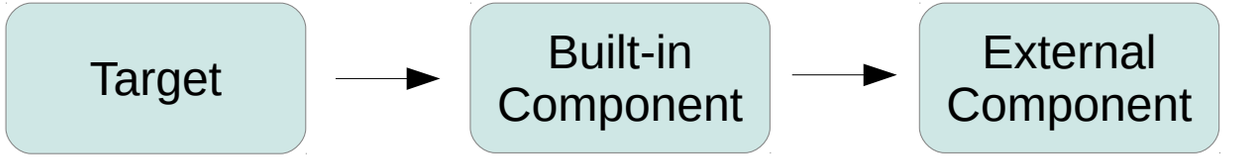
Step #3: Vuln. Assessment / Web App Assessment / Web Vuln. Scanner

```
"hackersh.org" \
    -> domain \
        -> dnsdict(“-4“) \
            -> nmap \
                -> nikto
```



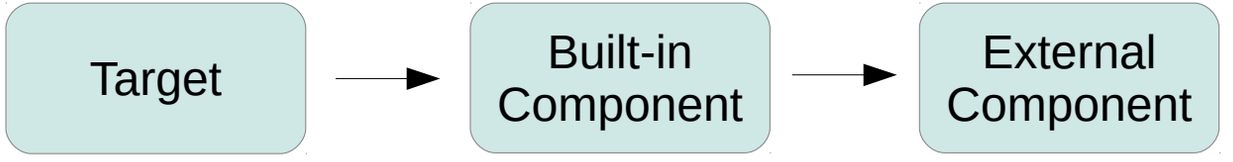
Step #4: Vuln. Assessment / Web App Assessment / Web Vuln. Scanner # 2

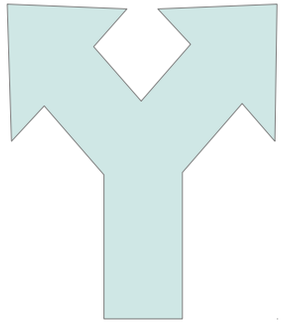
```
"hackersh.org" \
    -> domain \
        -> dnsdict(“-4“) \
            -> nmap \
                -> nikto \
                    -> w3af
```



Step #5: Reporting

```
"hackersh.org" \
    -> domain \
        -> dnsdict(“-4“) \
            -> nmap \
                -> nikto \
                    -> w3af \
                        -> print
```

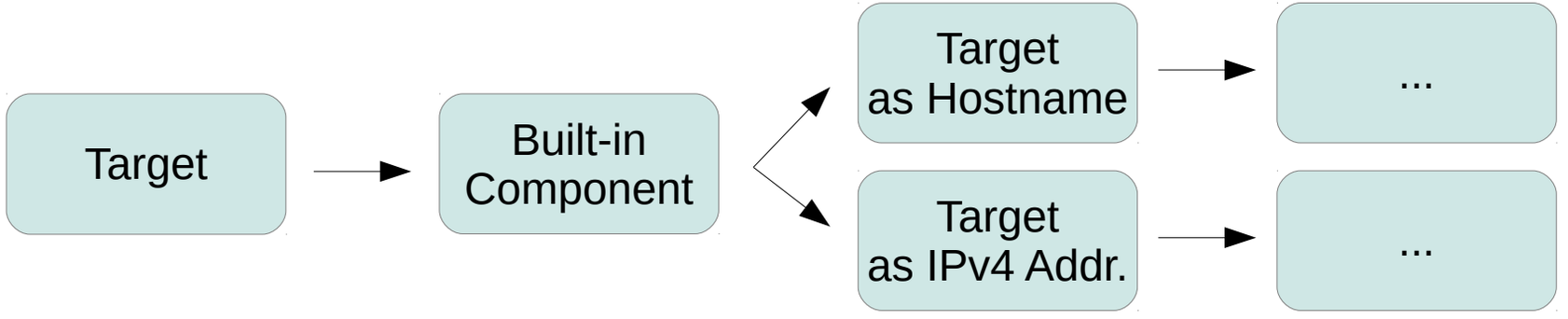




Fork

Target as Hostname + Target as IP

```
"ikotler.org" \
    -> hostname \
        -> [nslookup, pass] -> ...
```



Hackersh Roadmap

- Unit Tests
- Documention
- More Tools
  - Metasploit
  - OpenVAS
  - TheHarvester
  - Hydra
  - …
- Builtin Commands
- **<YOUR IDEA HERE>**

Questions?

Thank you!

My Twitter: @itzikkotler
My Email: ik@ikotler.org
My Website: http://www.ikotler.org

Pythonect Website: http://www.pythonect.org
Hackersh Website: http://www.hackersh.org

Feel free to contact me if you have any questions!