Now with more spiders!

PATHspider II: The Tutorial

Iain R. Learmonth
Electronics Research Group
University of Aberdeen

MAMI Summer School 2018 – Aberdeen, Scotland

mami
measurement and architecture for a middleboxed internet

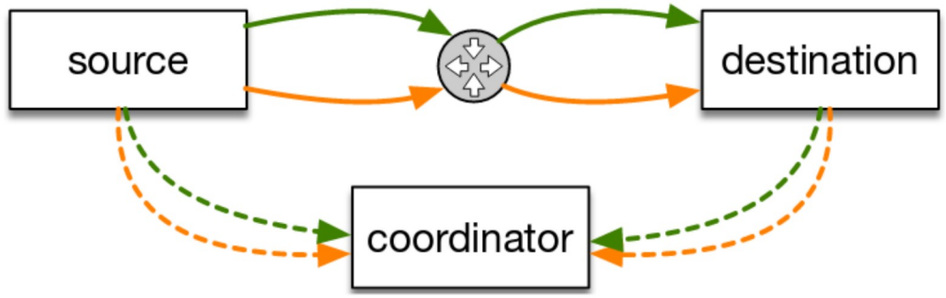
PATHspider

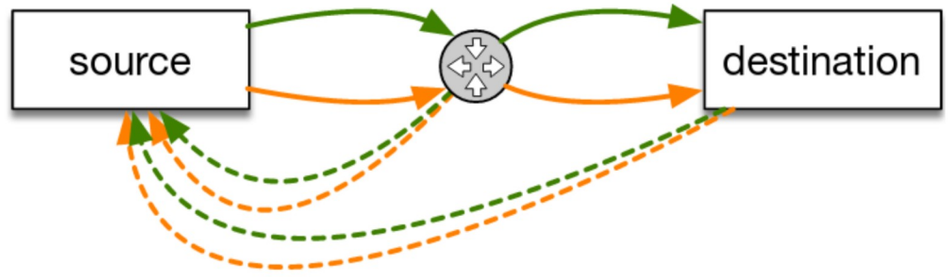
measurement | architecture | experimentation

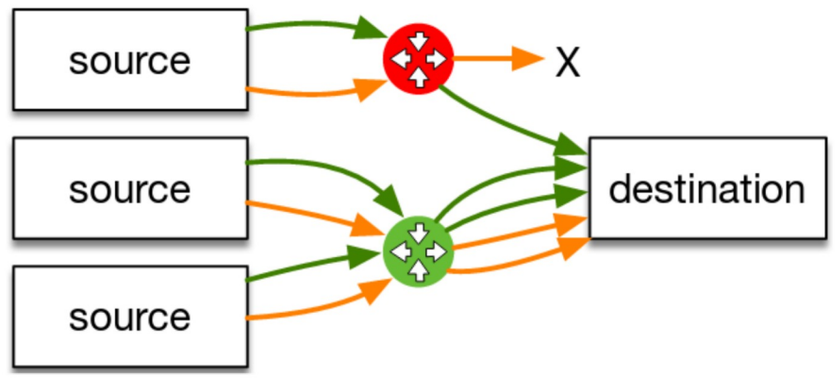
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421.The opinions expressed and arguments employed reflect only the authors' view. The European Commission is not responsible for any use that may be made of that information.

Active Measurement of Path Transparency

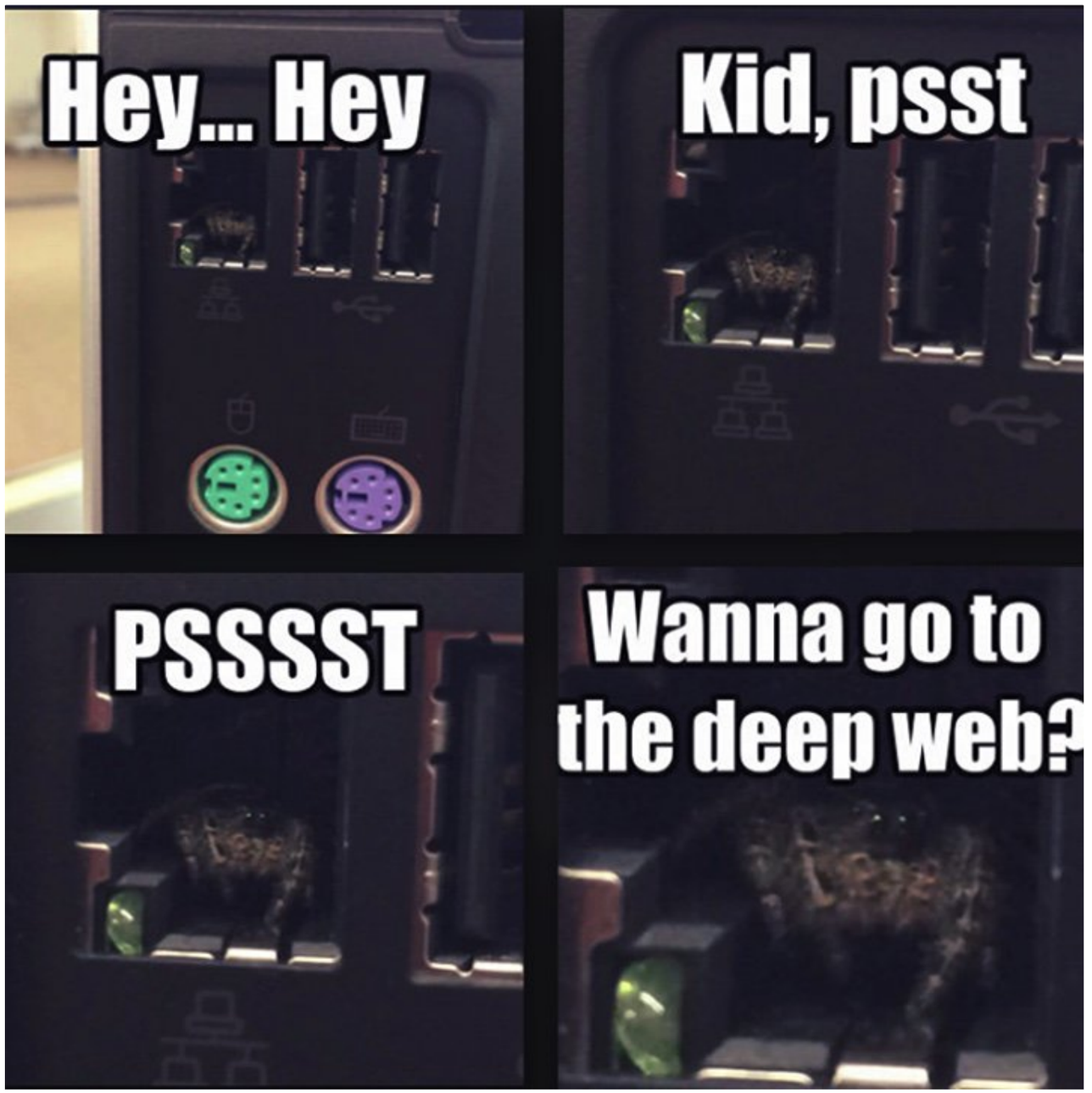
- Methodology:
  1. Throw packets at the Internet
  2. See what happens
- Ideal: two-ended A/B testing
- Scalable: one-ended A/B testing
- Multiple sources: **isolate** on-path from near-target impairment







Spiders in the network



ecnspider

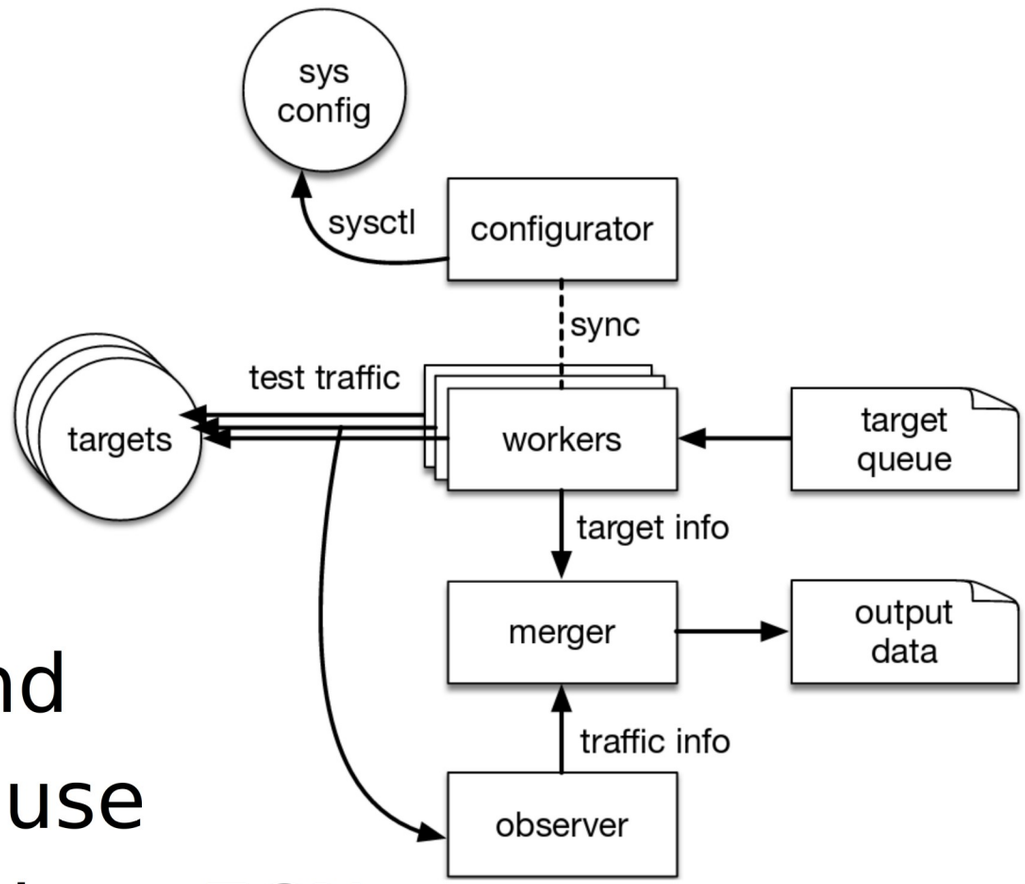
PATHspider before PATHspider

mami measurement

ecnspider

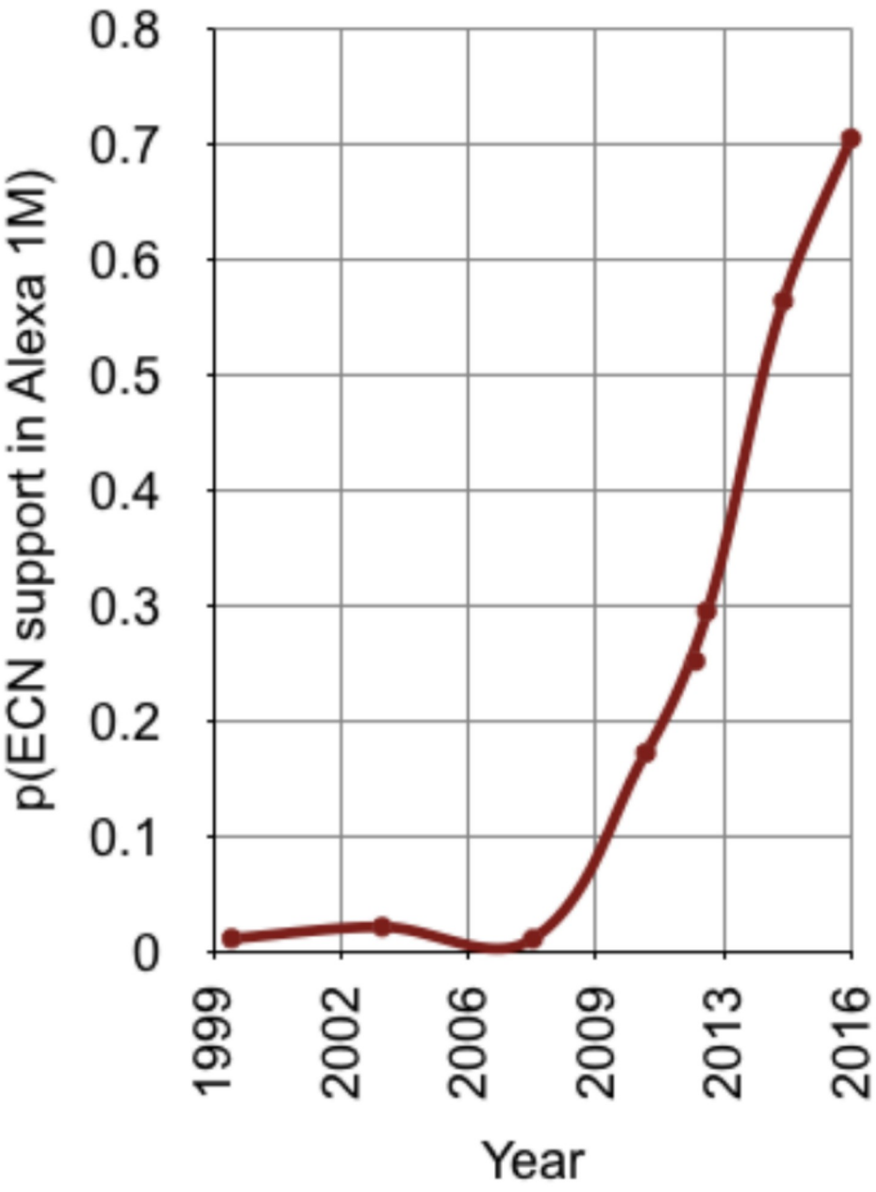
https://github.com/britram/ecnspider/

- The original implementation supported by mPlane/RITE
- Three distinct components:
  - DNS List Resolver
  - QoF Flow Meter
  - Active Traffic Generator
- Used hardcoded `sysctl(1)` and `iptables(1)` commands to cause packets to be emitted with various ECN-related flags



ecnspider Results

- **A measurement run in June 2016 from a single vantage point, a DigitalOcean server in Amsterdam, to the set of unique IPv4 and IPv6 addresses serving the top million websites, and found that 432544 of 617873 (70.005%) of IPv4 addresses and 20262 of 24472 (82.797%) IPv6 addresses will negotiate ECN**
- **This continues a trend ETH started observing in 2013**



PATHspider 1.0

More than just ECN

mami measurement

PATHspider 1.0

https://github.com/mami-project/pathspider/tree/1.0.1

- Architecture based closely on the original ecnspider
- Generalised to support more than just ECN
  - Added TCP Fast Open and DiffServ Codepoints
- Still performing A/B testing, but with more A/B tests
- Replaced QoF with a Python flowmeter implementation using *python-libtrace*
- Began to develop a generalised measurement methodology for path transparency testing
- Published at ANRW ‘16

Plugin Architecture

```
@implementer(ISpider, IPlugin)
class ExampleSpider(Spider):
    def config_zero(self):
        […] # Prepare the system for the A test (sysctl, etc.)
    def config_one(self):
        […] # Prepare the system for the B test (sysctl, etc.)
    def connect(self, job, pcs, config):
        […] # Perform the connection, open a socket
    def post_connect(self, job, conn, pcs, config):
        […] # Perform a post-connection operation, e.g a HTTP
GET
    def create_observer(self):
        […] # Create the observer, with the observer functions
    def merge(self, flow, res):
        […] # Merge connection results with flow results
```

Built-in Flowmeter

- PATHspider’s built-in flow meter is extensible via the plugin architecture
- Using *python-libtrace* to dissect packets, any flow property imaginable can be reported back based on the raw packets:
  - ECN negotiation (IP/TCP headers)
  - Bleaching of bits, dropping of options
  - Checksum recalculations
  - Number of packets containing a prime number of octets

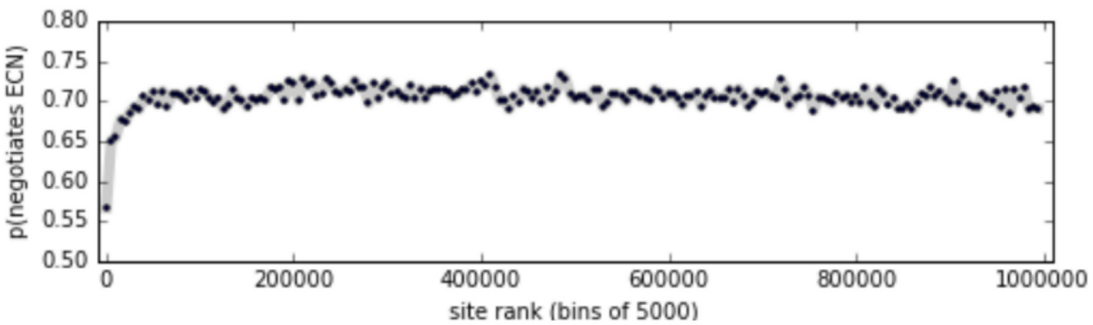
PATHspider 1.0 Results

Explicit Congestion Notification (ECN)

State of ECN server-side deployment, as measured from a Digital Ocean vantage point in Amsterdam on 13 June 2016:

	IPv4	IPv6	all
No ECN connectivity issues	**99.5%**	**99.9%**	**99.5%**
ECN successfully negotiated	**70.0%**	**82.8%**	**70.5%**

ECN negotiation by Alexa rank bin: note this is nearly uniform, but higher-ranked servers tend to disable ECN:



DiffServ Code Points (DSCP)

Initial study: 10006 out of 96978 **(10.31%)** of Alexa Top 100k websites had unexpected, non-zero DSCP values. More measurement is necessary to better characterize these anomalies.

TCP Fast Open (TFO)

Initial study: **330 IPv4 and 32 IPv6** addresses in Alexa Top 1M are TFO-capable (of which 278 and 28 are Google properties). DDoS prevention services, enterprise firewalls, and CPE tend to interfere with TFO. More measurement is necessary to analyze impairments.

PATHspider 2.0

n-dimensional hyperspider

mami measurement

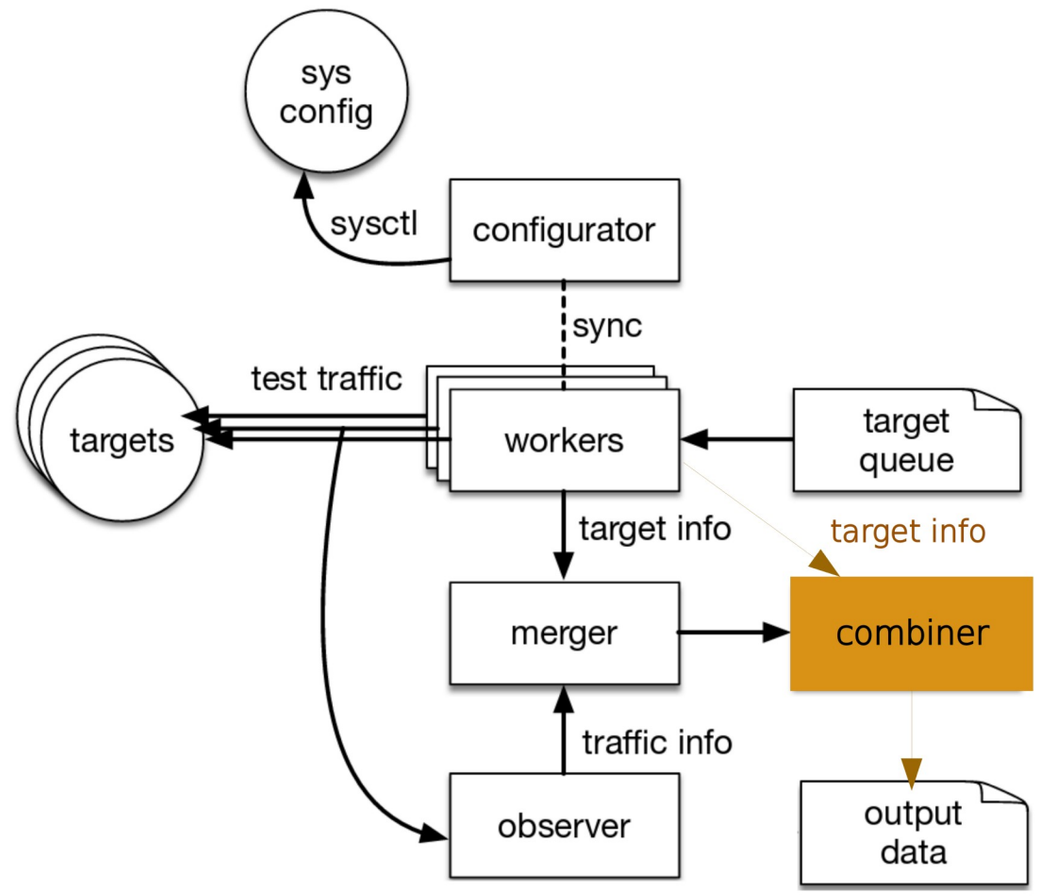
PATHspider 2.0

https://github.com/mami-project/pathspider/tree/master/

- Architecture changed to add a flow combiner
- Generalised to support more than just A/B testing
  - Any permutation of any number of tests
- Replaced PATHspider's HTTP code with *cURL*
- Added framework for packet forging based plugins using *Scapy*
- Completely rewritten (in Go) target list resolver
- Observer modules usable for standalone passive observation or analysis

A/B/.../n Testing

- Combiner thread holds a table of merged flows and waits for $|flows| == |tests|$
- Conditions generated based on the combined flows



Plugin Types

- Synchronised (traditional ecnspider)
  - ECN, DSCP
- Desynchronised (ecnspider with configurator disabled)
  - TFO, H2, TLS NPN/ALPN
- Forge (new in PATHspider 2.0!)
  - Evil Bit, UDP Zero Checksum, UDP Options

Connection helpers

- Instead of writing client code, use the code that already exists
- In the `pathspider.helpers` module:
  - DNS (dnslib)
  - HTTP/HTTPS (pycURL)
  - TCP (Python socket)
  - Tor HTTP/HTTPS (pycURL)
- For synchronised plugins, just use the helper
- For desynchronised plugins, the helpers are customisable, e.g. cURL helpers accept arbitrary CURLOPTs

Synchronised Plugin

```
class ExamplePlugin(SynchronizedSpider, PluggableSpider):
    name = "..."
    description = "..."
    version = "..."
    chains = [BasicChain, ...]
    connect_supported = ["http", "https", "tcp", "dnstcp"]

    def config_zero(self): # e.g. sysctl
        ...
    def config_one(self): # e.g. sysctl
        ...
    configurations = [config_zero, config_one, ...]
    def combine_flows(self, flows): # generate conditions
        ...
        return conditions
```

Desynchronised Plugin

```
class ExamplePlugin(DesynchronizedSpider, PluggableSpider):
    name = "..."
    description = "..."
    version = "..."
    chains = [BasicChain, …]
    connect_supported = [“tcp”, ...]

    def connect_zero(self): # e.g. call http helper
        ...
    def connect_one(self): # e.g. call http helper
        ...
    connections = [connect_zero, connect_one, ...]
    def combine_flows(self, flows): # generate conditions
        ...
        return conditions
```

Forge Plugin

```
class ExamplePlugin(ForgeSpider, PluggableSpider):
    name = "..."
    description = "..."
    version = "..."
    chains = [BasicChain, …]
    connect_supported = [“tcp”, ...]

    def forge(self, job, seq): # e.g. make a tcp syn
        ...

    def combine_flows(self, flows): # generate conditions
        ...
        return conditions
```

Observer modules

- While these used to be part of plugins in PATHspider 1.0, they are now independent and so can be reused
  - BasicChain, DNSChain, DSCPChain, ECNChain, EvilChain, ICMPChain, TCPChain, TFOChain
- These can also be used together, limiting each chain to just a single layer and letting the combiner produce conditions.
- Chains can produce information to be consumed by other chains later in the list.
- These can now be run independently of a PATHspider measurement:
- e.g. > `pspdr observe tcp ecn`

Target List Resolution

https://pathspider.net/hellfire

- Hellfire is a parallelised DNS resolver. It is written in Go and for the purpose of generating input lists to PATHspider, though may be useful for other applications.
- Can use many sources for input:
  - Alexa Top 1 Million Global Sites
  - Cisco Umbrella 1 Million
  - Citizen Lab Test Lists
  - OpenDNS Public Domain Lists
  - Comma-Seperated Values Files
  - Plain Text Domain Lists

Hands-on Time!

Emit some packets

mami measurement