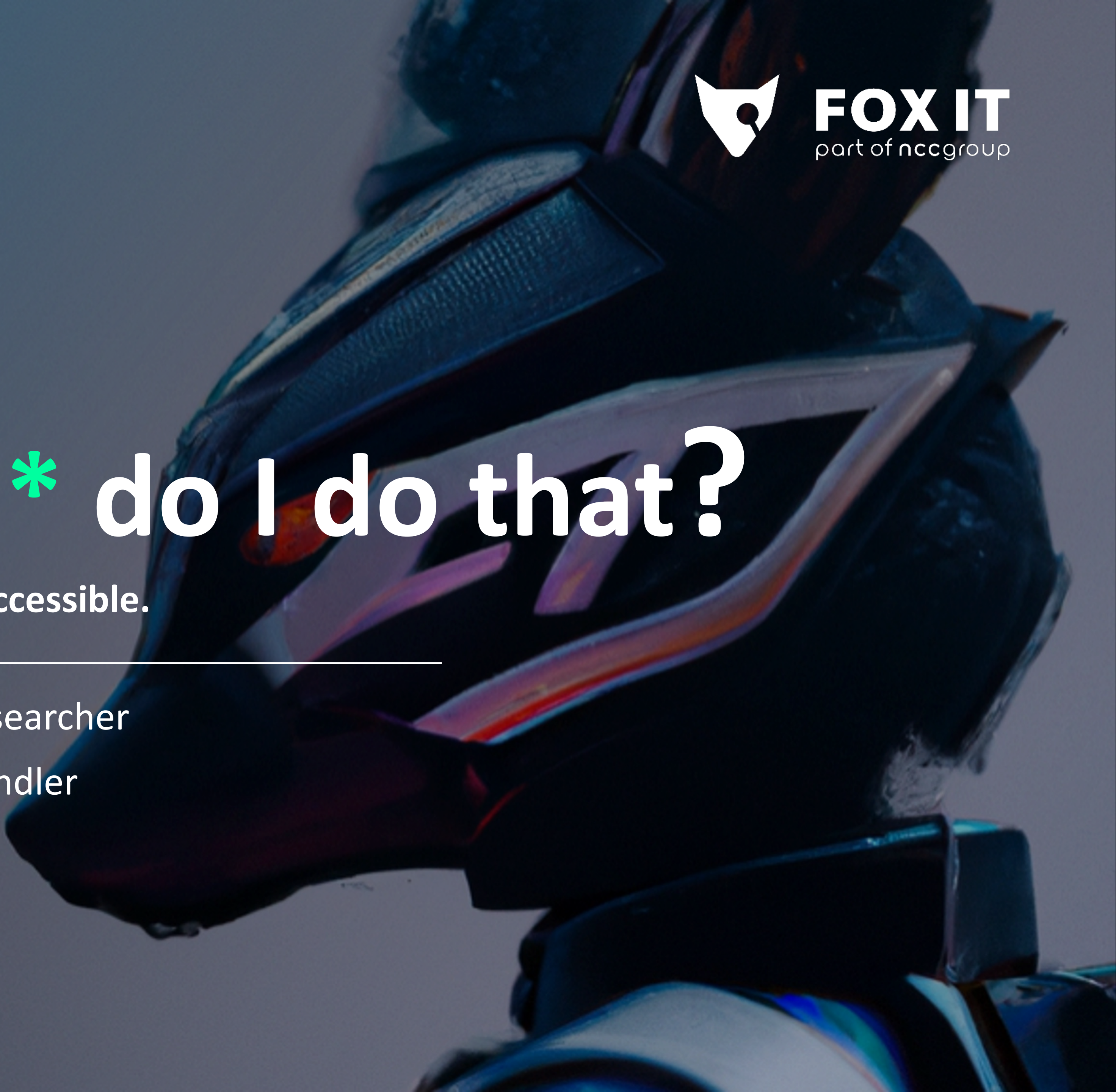
FOSDEM'26

FOX IT part of nccgroup



How the **** do I do that?

Making 300+ forensic parsers easily accessible.

Erik Schamper Security Researcher

Lennart Haagsma Incident Handler

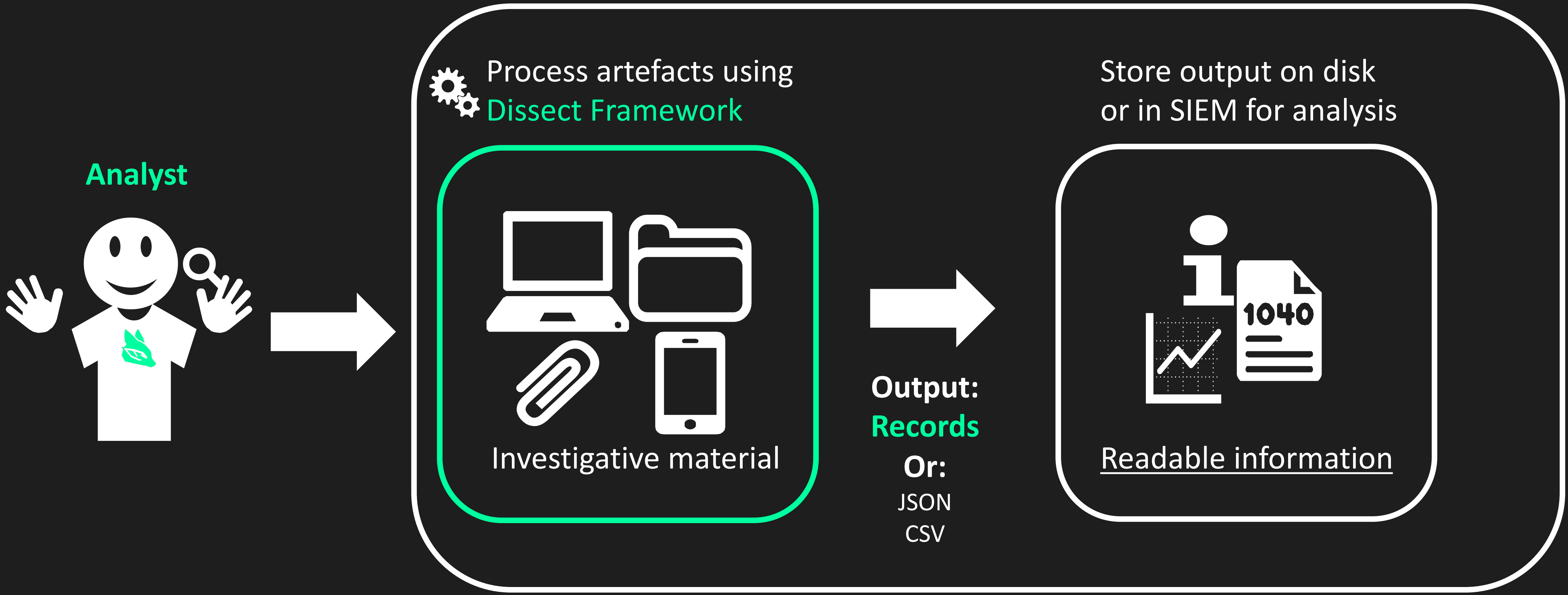
2026-02-01 TLP:CLEAR

Dissect is a pure Python, no dependencies, open source, forensic investigation framework.

Goal: *Parse any high-level artefact at scale no matter the container it resides in.*

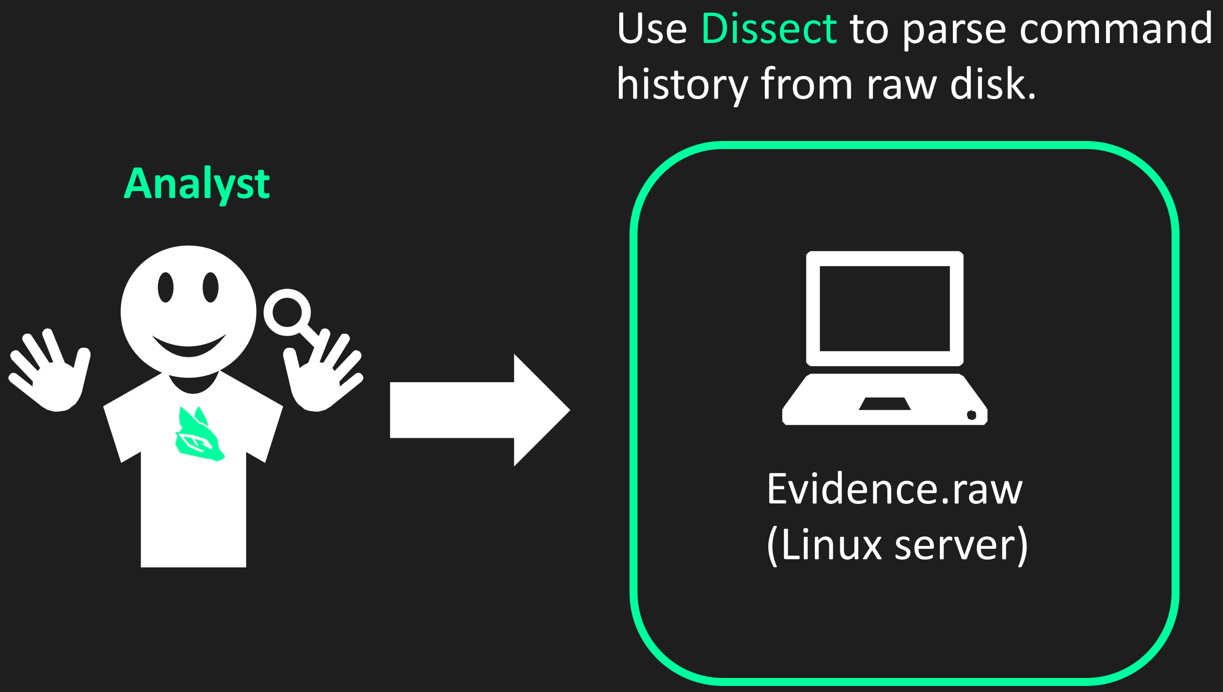
Developed by Fox-IT (part of NCC group) and used (and contributed to) by government, cybersecurity firms and other private organizations.

Introduction Dissect: workflow



Introduction Dissect: usage

```
$ target-query -f commandhistory Evidence.raw
```


Use Dissect to parse command history from raw disk.
Analyst
Evidence.raw (Linux server)

Dissect will:

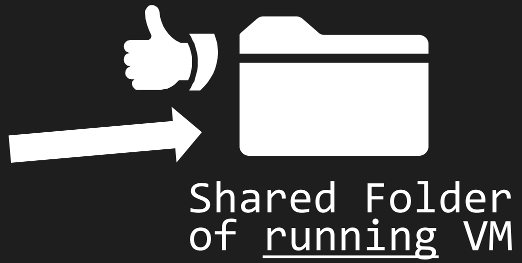
- ➢ Identify the target **.raw file** as a **full disk image**.
- ➢ Identify and read the **EXT4** filesystem
- ➢ Identify a Linux-like operating system
- ➢ Run the commandhistory plugin
  - ➢ Iterate user accounts for Linux /etc/passwd
  - ➢ Look for common command history files in home directories of users
  - ➢ Yield record for each line in found history file

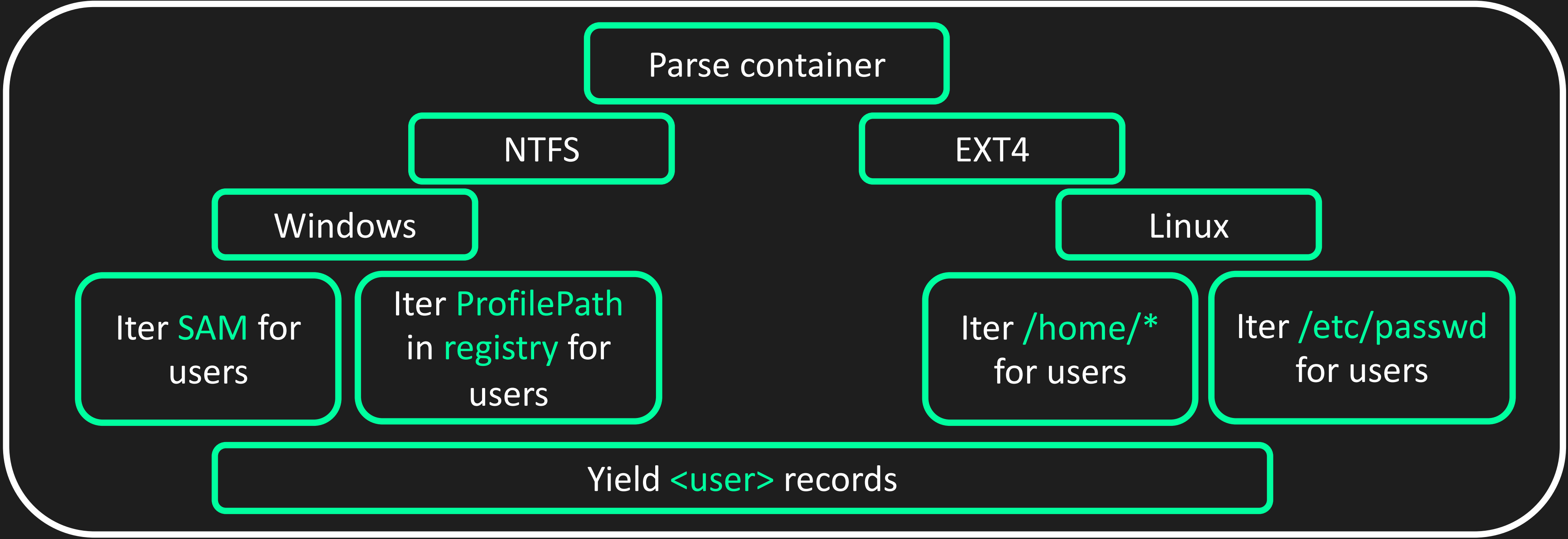
```
> <unix/history hostname='ubuntu', domain=None,
  ts=None, order=0, command='whoami', shell='bash', source='/root/.bash_history',
  user_id='0', user_group='0', user_home='/root/'>
```

Bringing “it just works” to DFIR

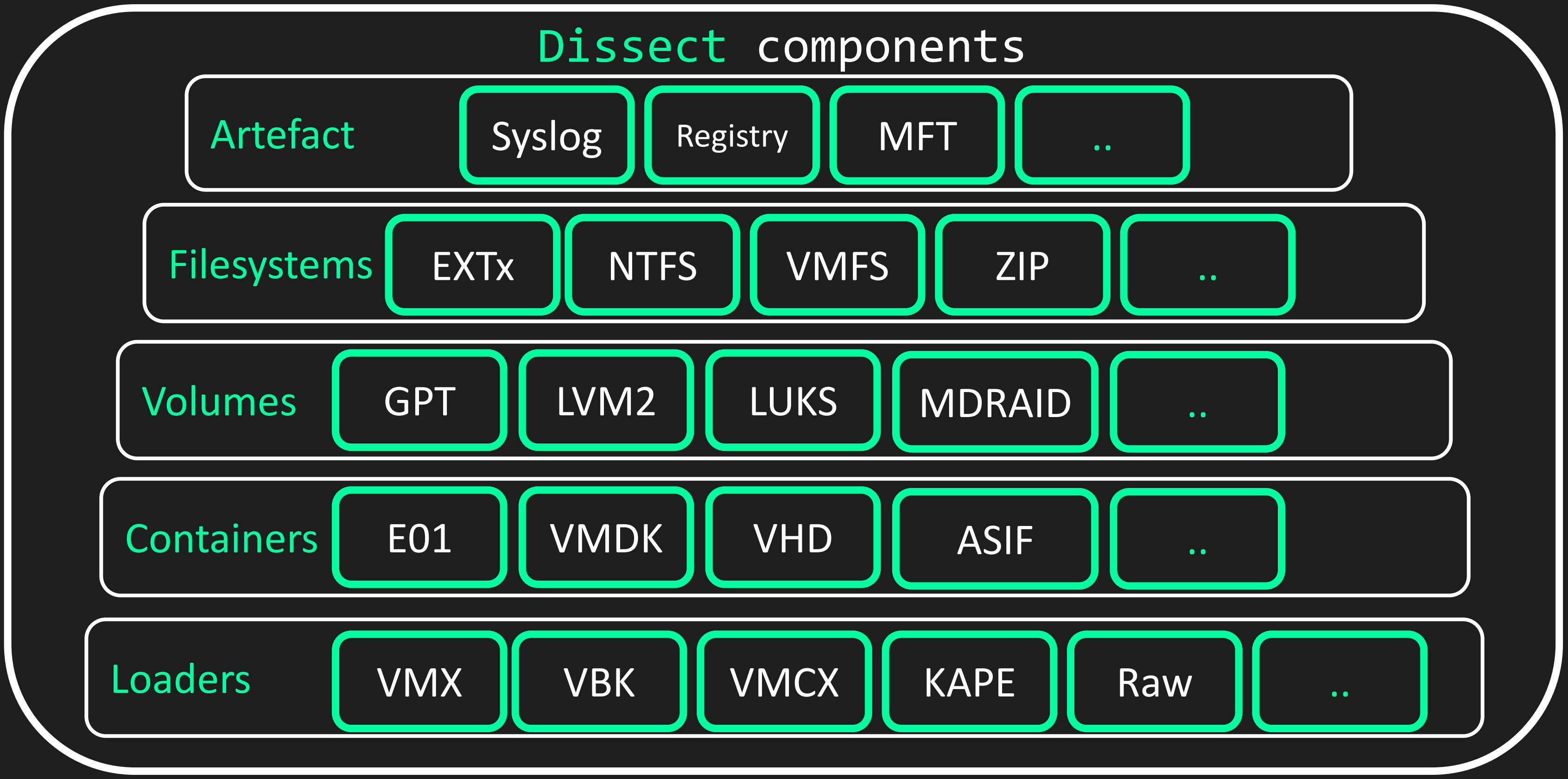
```
from dissect.target.target import Target

t = Target.open("/path/to/evidence.{vmdk,tar,dd,..}")
list(t.users())
```





Dissect under the hood



Dissect plugins are the parsers that process the data.

Dissect loaders are the glue that makes it all work together.

```
$ target-query -f <artefact> /systems/*
```

FOX IT
part of nccgroup

```
$ target-query -f <artefact> /systems/*
$ target-query -f ... *.E01
$ target-query -f ... *.vmx / *.vmdk
$ target-query -f ... *.vmcx / *.vhdx
$ target-query -f ... *.vma
$ target-query -f ... smb://...
```

FOX IT
part of nccgroup

```
$ target-query -f ... \
  | rdump -w json://
```

```
$ target-query -f ... \
  | rdump -w csv://
```

FOX IT
part of nccgroup

```
$ target-query -f ... \
| rdump -w splunk://<ip>:1337
```

```
$ target-query -f ... \
| rdump -w elastic://<ip>:1337
```

What you get

Collection

- `acquire`

Cool stuff

- `target-diff`
- `target-info`
- `target-mount`
- `target-query`
- `target-shell`

Also cool

- `target-dd`
- `target-fs`
- `target-inspect`
- `target-qfind`
- `target-reg`
- `target-yara`

FOX IT
part of nccgroup

Dissect ‘children’ or child-plugins

Targets can have sub-targets, referenced within Dissect as ‘children’.

Example of a child is a Docker-container running on a target that runs Docker as a ‘hypervisor’.

Hypervisors are transparent

Artifacts become reachable through (*multiple layers of*) hypervisors.

A child-plugin is just another layer to parse through, like a file-system or container format.

Supported ‘Hypervisors’

Dissect has support for various ‘hypervisors’, example of these are:

```
[Hyper-v, VMWare {workstation, player, ESXi},
Virtualbox, Proxmox, Docker, ...]
```

And some Dissect doesn’t support (yet):

```
[XenServer, LXC/LXD, ...]
```

Child arguments ;-)

All the target-{query, fs, shell} tools support the following child arguments:

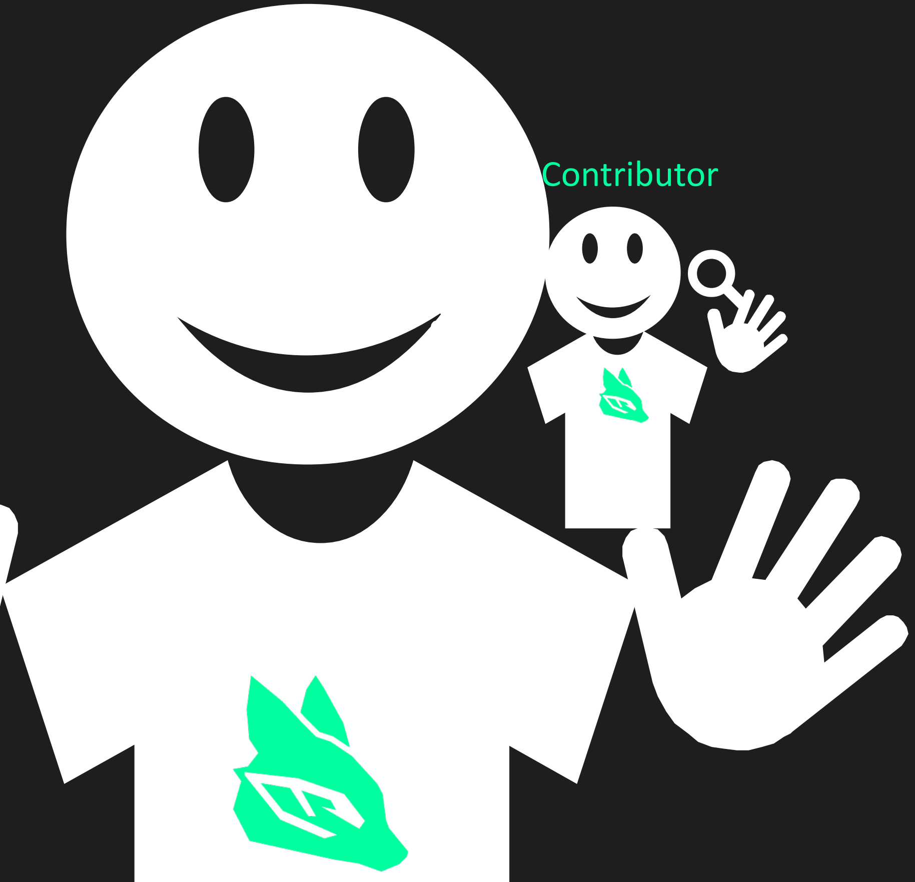
```
--list-children # prints a list of identified children
--children      # runs command on all child-targets
--child         # sets specific child-target as the new target
--recursive     # makes --list-children and --children recursive
```

```
# Example of child targets:
[Windows Target]
   [Windows Subsystem for Linux (WSL)]
      [Docker container, …]
   [VMWare Workstation Virtual Machine, …]
```

FOX IT part of nccgroup

Demo: Matryoshka-image

```
 _________________________________________
< Standing on the shoulders of giants.txt >
 -----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```



Commit History for dissect.target/dissect/target/plugins/child on main
~10 authors from multiple organizations since jul 2022

```
$ cat giants.txt
[“cecinestpasunepipe”, “ JSCU-CNI”, “lhaagsma”,
“martinvanhensbergen”, “Miauwkeru”, “otnxSl”, “pyrco”,
“Schamper”, “sulonl, “ Zawadidone”]
```

```
root@pve:~/dev/fox# 
```

```python
from dissect.target.target import Target

_, c = list(Target.open("local").list_children())[6] # Get info about 'parent' from 'local' pve.
# <target/child hostname='pve' domain=None type='proxmox' name='Never' path='/etc/pve/qemu-server/106.conf'>

t = Target.open(str(c.path))
index = '<parent>'

print(f"{index:<17} type: {c.type:<19} name: {c.name:<6} os: {t.os:<8} {t.version:<48} hostname: {t.hostname}")

for index, c in t.list_children(recursive=True):
    ct = t.open_child(index)
    print(f"{index:<17} type: {c.type:<19} name: {c.name:<6} os: {ct.os:<8} {ct.version:<48} hostname: {ct.hostname}")
```

FOX IT
part of nccgroup

```python
from dissect.target.target import Target

_, c = list(Target.open("local").list_children())[6] # Get info about 'parent' from 'local' pve.
# <target/child hostname='pve' domain=None type='proxmox' name='Never' path='/etc/pve/qemu-server/106.conf'>

t = Target.open(str(c.path))
index = '<parent>'

print(f"{index:<17} type: {c.type:<19} name: {c.name:<6} os: {t.os:<8} {t.version:<48} hostname: {t.hostname}")

for index, c in t.list_children(recursive=True):
    ct = t.open_child(index)
    print(f"{index:<17} type: {c.type:<19} name: {c.name:<6} os: {ct.os:<8} {ct.version:<48} hostname: {ct.hostname}")
```

```
root@pve:~/dev/fox#
```

```
root@pve:~/dev/fox# 
```

Code is the product

- Library code is first class citizen, CLI tools are secondary
  - Challenge: contributions need a lot of attention
- Centered around familiar Python APIs
  - Prioritize familiarity instead of reinventing the wheel
- Promotes custom tool development

target-shell --python

```
In [1]: t.hostname
Out[1]: 'dissect-centos'

In [2]: t.version
Out[2]: 'CentOS Linux 8'

In [3]: t.fs.path("/etc/hosts").read_text()
Out[3]: '127.0.0.1   localhost localhost.localdomain localhost4 localhost4.lo
caldomain4\n::1         localhost localhost.localdomain localhost6 localhost6
.localdomain6\n'

In [4]: from dissect.cstruct import dumpstruct

In [5]: dumpstruct(t.filesystems[1].get("/etc/hostname").entry.inode)

00000000  49 4e 81 a4 03 02 00 00  00 00 00 00 00 00 00 00   IN..............
00000010  00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000020  64 47 8b 1f 3a 60 bc 6c  64 47 97 e9 13 1b 6f 80   dG..:`.ldG....o.
00000030  64 47 97 e9 13 1b 6f 80  00 00 00 00 00 00 00 1b   dG....o.........
00000040  00 00 00 00 00 00 00 01  00 00 00 00 00 00 00 01   ................
00000050  00 00 23 01 00 00 00 00  00 00 00 00 b2 53 39 d2   ..#..........S9.
00000060  ff ff ff ff f5 6d 20 0e  00 00 00 00 00 00 00 09   .....m .........
00000070  00 00 00 07 00 00 16 5c  00 00 00 00 00 00 00 00   .......\........
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000090  64 47 86 f2 1c ab 7a 40  00 00 00 00 00 23 54 f4   dG....z@.....#T.
000000a0  80 eb e1 39 92 7b 46 b3  ae 91 eb b5 0c 3a 09 77   ...9.{F......:.w

struct xfs_dinode:
- di_magic: 0x494e
- di_mode: 0x81a4
- di_version: 0x3
- di_format: 0x2
- di_onlink: 0x0
- di_uid: 0x0
- di_gid: 0x0
```

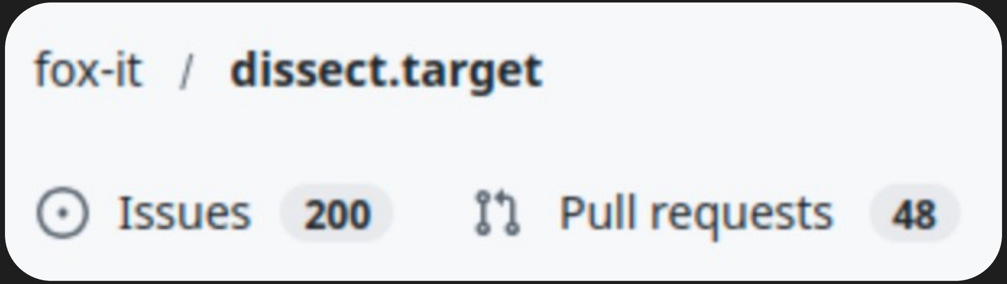
FOX IT
part of nccgroup

Open-source!

- Since 4th of October 2022!
- Open-source? For a more secure society!
- High adoption rate, number of contributors picking up
  - Government, competitors, students, researchers
- Incredible, high-quality contributions
  - DPAPI decryption, Velociraptor support, improved Linux support, new parsers

New challenges

- Rate of finding bugs 5x
  - Lots more users = more exposure
- Rate of fixing only 1.2x…
  - Not everyone is a suitable (core) contributor
- How to collaborate more effectively?
  - Tips & tricks are welcome


fox-it / dissect.target

Issues 200 Pull requests 48

Tool verification

Tool verification is hard

- How to keep up with OS and tool updates?

Everyone does their own(?)

- Open-source effort for tool verification?

Please involve us if you're working on this

Community use-cases

Besides forensics / blue team stuff…

- Investigating actor infrastructure at scale
- OT security + compliance insights at scale

Actor tracking at scale

- Put the focus on analysis, skip the "boring" stuff
  - Immediate actionable access from any source data
- Uniformity in methodology, results and reporting
  - Enforces standardization in analyst tooling too
- Framework and Python is accessible for analysts
- Custom pipelines and analysis scripts for actor tools
  - Centralized knowledge, usable on all current and future source data

OT insights at scale

Limited ability to run software on OT devices

- Old hard- and software, stability concerns, certification/warranty, connectivity
- Big % of fleet

Still need visibility

- (Cyber) security monitoring, compliance

Current solution: a guy in a golf cart

OT insights at scale

Creative problem solving

Backups are created for disaster recovery

… backups can be inspected with Dissect!

Link backup creation to a processing workflow

Parse all relevant artefacts for security and compliance

Security logs, USB history, …

Insights within minutes, rather than driving around campus for hours

>Ξ Takeaways

- Dissect as your central processing framework
  - Consistent quality and verifiable
- Reusability of tools on any source material

- … and more we don't have time for!
  - Transparent analysis on FDE
  - Mobile and appliance analysis
  - Red Team use-cases

Thank you!

```
$ pip install dissect
```

Get involved:

fox-it/dissect

dissect@fox-it.com

FOX IT
part of nccgroup