Windows Disk Image

- Assumes basic setup from setup
- All command below should be
  - copied to a “IR log” text/md file
  - edited as needed
  - executed with output copied back to “IR log”
    - recommendation: move commands/output from “useless commands” to second “IR dump” text/md file
- ## is a placeholder for a number

setup

```
echo 'source $HOME/.case.env' >> ~/.bashrc
```

Env variables

```
rm ~/.case.env
pwd
WD=/case/working/folder && echo "WD=$WD" >> ~/.case.env
E=/case/working/folder/evidence/evidence.E01 && echo "E=$E" >> ~/.case.env
```

Disk Image basics

```
mmls -r $E
sudo -E ~/.local/bin/imount -p -o ## -md /mnt/imount $E
# -p has 'pretty' & predictable folder name, fails if in use
# new tab or crtl-z
DISK_C=<imount_dir> && echo "DISK_C=$DISK_C" >> ~/.case.env
```

Hayabusa

```
target-fs "$E" cp 'c:/Windows/System32/winevt/Logs' --output ./logs
docker run -it --rm -v "$PWD":/work --entrypoint /opt/hayabusa/hayabusa
tabledevil/hayabusa:3.8.1 csv-timeline -d /work/logs -o
/work/output/hayabusa.csv -p super-verbose
docker run -it --rm -v "$PWD":/work --entrypoint /opt/hayabusa/hayabusa
tabledevil/hayabusa:3.8.1 json-timeline -d /work/logs -o
/work/output/hayabusa.jsonl -p super-verbose
```

Plaso

```
docker run -it --rm --entrypoint=/bin/bash -v ./:/work log2timeline/plaso
```

```
# docker run -it --rm --user :$(id -g) --entrypoint=/bin/bash -v ./:/work
log2timeline/plaso

# one step
psteal.py --source /work/evidence/<filename> -o dynamic,json_line -w
/work/data/plaso_#.json

# two step
log2timeline.py --storage-file /work/data/timeline.plaso
/work/evidence/<filename>
psort.py -o json_line -w /work/output/plaso_out.json
/work/data/timeline.plaso
#psort.py -o json_line -w /work/output/plaso_out.json
/work/data/timeline.plaso


cp output/plaso_out.json tools/splunk/import/
cd tools/splunk
docker compose up -d
# -> localhost:8000, admin:password,
settings->add->monitor->files->index_once->source_type=plaso
```

Splunk

```
mkdir -p etc/system/local/
vim props.conf
cp props.conf etc/system/local
```

Browser

```
Users\...\AppData\Local\Google\Chrome\User Data\Default\History
Users\...\AppData\Local\Microsoft\Edge\User Data\Default\History
Users\...\AppData\Roaming\Mozilla\Firefox\Profiles\<ProfileName>.default\pla
ces.sqlite


C_DR=
EUSER=
mkdir -p evidence/chrome
mkdir evidence/edge
mkdir evidence/firefox
cp -r $C_DR/Users/$EUSER/AppData/Local/Google/Chrome/User\
Data/Default/History evidence/chrome
cp -r $C_DR/Users/$EUSER/AppData/Local/Microsoft/Edge/User\
Data/Default/History evidence/edge
cp -r $C_DR/Users/$EUSER/AppData/Roaming/Mozilla/Firefox/Profiles/*-
release/places.sqlite evidence/firefox/
```

```
sqlitebrowser evidence/firefox/places.sqlite
sqlitebrowser evidence/chrome/History
```

PE

```
https://github.com/sethenoka/Install_EZTools.git
```

Prefetch on Linux

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
source ~/.cargo/env
rustup toolchain install 1.85.0
rustup default 1.85.0
```

```
use anyhow::Result;
use prefetch_core::parse;

fn main() -> Result<()> {
    let path = std::env::args()
        .nth(1)
        .expect("Usage: pf-analyzer <file.pf>");

    let bytes = std::fs::read(path)?;

    let pf = parse(&bytes)?;

    println!("Executable : {}", pf.executable_name);
    println!("Run count  : {}", pf.run_count);

    println!("Last run times:");
    for ts in &pf.last_run_times {
        println!("  {:?}", ts);
    }

    Ok(())
}
```

```
use prefetch_core::parse;

fn main() -> anyhow::Result<()> {
    let dir = r"C:\Windows\Prefetch";

    for entry in std::fs::read_dir(dir)? {
        let path = entry?.path();

        if path.extension().and_then(|x| x.to_str()) != Some("pf") {
            continue;
        }
```

```
        let bytes = std::fs::read(&path)?;

        match parse(&bytes) {
            Ok(pf) => {
                println!(
                    "{} | runs={} | last={:?}",
                    pf.executable_name,
                    pf.run_count,
                    pf.last_run_times.first()
                );
            }
            Err(e) => {
                eprintln!("{} -> {}", path.display(), e);
            }
        }
    }

    Ok(())
}
```

Extended disk image anaylsis

```
fsstat -o <offset> $E
istat -o <offset> $E 5 # root node
istat -o <offset> $E <inode from fsstat>

fls -o <offset> -m C: -r $E > data/bodyfile
mactime -b data/bodyfile -d -z UTC yyyy-mm-ddThh:mm:ss >
output/disk_timeline.csv

mactime -b data/bodyfile -d -z UTC yyyy-mm-ddThh:mm:ss..yyyy-mm-dd >
output/disk_timeline.csv


# dissect

target-query -f hostname,domain,version,ips,install_date,timezone $E
# much more useful for queries on multiple disks at once

target-query -j -f services $E | jq -r '.name'
# JSON output → jq
target-query --list | grep -iE
'userassist|shimcache|amcache|services|powershell_history|browser.history'
```