DEFENCIA · KNOWLEDGE BASE

Stop Bad Things

A compiled handbook of DFIR, digital forensics, malware analysis and information-security governance — condensed from defencia.dk.

Edition 2025 · Compiled for print · defencia.dk

Contents

DEFENCIA

Part I · Incident Response & Process

Emergency Handling & DFIR

What do you do if your company is hit by a cyber attack? This is the starting point: how to prepare, what to have ready, and how to begin handling and reporting an incident.

PREPARATION FIRST RESPONSE PROCESS

→Where to start

No security in place at all? Then call a friend — that advice is meant 100% seriously. You would not start an expedition to a mountain top from day one without preparation.

- Find out what your business depends on most — e.g. your website; without it there is no sale.
- The CIS Controls are a good place to start; learn to do a risk assessment (CIS-RAM) and follow their recommended controls.
- Look at the CREST procurement guide for dealing with cyber attacks — you need a basic plan whether your full plans are in place or not.

> Think of it like investing in a fire blanket and extinguisher even before you have escape plans and alarms — a basic readiness beats none.

→What to have ready (physically & practically)

- Overall IT security plan + Acceptable Use Policy (what users may and may not do).
- Emergency management for crashes and security incidents — define when something *is* a security event.
- Documents with phone numbers and an escalation schedule, to minimise doubt.
- A **war room** and a clear list of who handles what.
- Involve the DPO early on personal-data loss; keep contact with the data protection authority.
- A secured room where all data is collected, accessible only to authorised personnel (insider-threat aware).
- Task distribution — including food, drink and logistics, which are easy to overlook but matter a lot during long incidents.

→Preparing the digital toolbox

- Hardware: laptop, USB storage, hard drives.
- Software: virtualization, screenshots, OSINT tools for securing online data.
- A jump-bag (see the dedicated page).
- An analysis lab for malware, logs and network traffic.

→Write-block capability

To secure evidence correctly you must avoid writing to the disk you are securing — otherwise you contaminate the evidence. Write protection can be achieved in software or hardware.

Approach	How	Cost
Software (live boot)	Boot from USB with CAINE or Paladin — read-only by design.	Free
Software (installed)	Tools like Safe Block write-protect any device you connect (SATA + USB in one).	Paid, cheaper than hardware
Hardware	Physical device inserted between disk and computer; works over USB, no real speed loss on USB 3.x.	One-time investment (~3,000–8,000 DKK)

> Examples: Weibetech FUD (reads serial/product name), and Tableau from Guidance Software at the higher end. **Buy what fits YOUR task — you are building YOUR toolbox.**

The Phases

The phases are the periods a company has to move through to handle an attack — from peace-time preparation to lessons learned, based on the NIST (4) and SANS (6) models.

NIST SANS LIFECYCLE

→The incident phases

The phases are the periods a company moves through to handle an attack — outlined here from experience and the textbook. NIST uses 4 phases; SANS uses 6.

Phase	What happens
Preparation	Peace-time. Prepare tools (hardware + software) and all documents — governance, IT security policy, contingency and crash plans. The link to governance lives here.
Identification & Analysis	Something goes wrong. Triage whether it is a security incident or an ordinary crash. Collect data, identify how the attacker got in, analyse malicious file behaviour, domains and IPs contacted.
Containment (Lockdown)	Use what you have learned to block the attack — block domains/IPs in DNS, firewall, proxy; block files in anti-malware; detect by hash; alert on patterns with YARA; close shared drives to stop spread.
Recovery (Restore)	Identify **day zero** — the date the incident began — so you can restore from before that point.
Monitoring	For larger attacks, run an intensified monitoring period (days to months). If something recurs, lock the environment down again to stop re-infection.
Back to normal (Lessons Learned)	Resume normal operations, hold a retrospective, capture what went well or badly, and feed it into the plans — driving Continual Service Improvement.

→Going back is allowed

> If you discover something was missed, or new data appears, nothing stops you from going back and re-analysing the malware or artifact. The phases are a cycle, not a one-way street.

Emergency Management

Handling bad situations starts with knowing your own capabilities and aligning expectations with management — then preparing for the worst and creating a clear overview.

TRIAGE READINESS

→Know your capabilities

One of the most important first steps is to know your capabilities and align expectations with management. Some tasks — forensics, malware reverse engineering — need special, expensive skills that many organisations choose to hire in.

> Even without those capabilities, you can almost always perform a **triage**: an introductory investigation over a set window (e.g. 2–4 hours) before handing off to a third party. You can still answer a lot of questions in that time.

→Prepare for the worst

- What is the worst that can happen to your business?
- Which systems hold log data? (cloud, network gear, computers, websites…)
- How do you collect that data? (Police will ask for data that can recreate the scenario.)
- Can you recover data — are backups in place and do they actually work?
- What tools do you want ready, and where are they kept?

→Create an overview

- Define who does what when something goes wrong — war room, board with task owners.
- Who owns the incident, and who communicates what is happening?
- Distribute responsibility for data collection; in large companies appoint a data manager so data is stored and described correctly with timestamps.
- What data is compromised?

> Should authorities/Police be contacted? If so, get them on board **early** — in Denmark you can request an IT contact from NC3, present in every police district.

Important Questions

During an attack, the right questions provide crucial insight. A structured checklist for interrogating the artifacts, the users, the locations and the timeline.

CHECKLIST INVESTIGATION

→Questions to ask during an attack

The right questions provide important insight into how to get out of trouble. What happened, and what indicators do we have?

→What were the artifacts used for?

- Have the files been run? How did they get in? With what input/output?
- Created as a service? Set to autorun (enabled or disabled)?
- Were the artifacts removed afterwards, or are they freely available?
- Do we know what the files do? (test in a sandbox)

→Who used the files?

- Local, domain or remote user? What rights?
- Are there domain admins at all?
- Who ran the files, and at what privilege level (admin vs normal user)?

→Where were the artifacts found?

- Shellbags, MRU, Event Logs, Services, MFT, NTFS?
- Memory? Data streams?
- Payload from other connected IPs/URLs? Path to file / registry.

→When are the artifacts from?

- Which data layers were timestamps taken from? Where are the remaining timestamps?
- Do we trust the timestamps, or are they obfuscated by malware?

→What did we select, and why?

- How did we classify files as interesting? What filtration method?
- Which search criteria and analysis methods?

Jump-bag

Plan your kit so it can be used the moment it is needed. Everything in one bag reduces the time to get started — and the contents must stay there in peace-time.

FIELD KIT READINESS

→The purpose

Assemble your kit so everything you need is in one place — this cuts the time to get started. The items must stay in the bag in peace-time, or it loses its value exactly when you need it. It also holds printed documentation: contingency plans, contact info for vital partners, and incident-handling procedures.

→Contents

No fixed list exists — this is what the author wants in the bag.

- **Pen & paper** — the most overlooked yet most important; for notes, times and who you spoke to.
- Printed contingency plans, partner contacts, incident procedures.
- Hard drives (various sizes) and USB keys for copying/handing over data.
- Tool set (ideally with a magnetic tray / iFixit lid), antistatic bracelet, multitool, headlamp.
- Write protection — USB with CAINE Linux, plus hardware or software write-blocker.
- A computer, preferably Linux, for dealing with malware; optionally one with the company image to test AV definitions. Remember the power supply!
- Water and food (muesli bar) — not trivial on multi-hour jobs; personal items like deodorant/toothbrush.
- Cables: network, USB (A→micro/C), USB→SATA, SSD/NVMe/M.2 adapters; multi-format card reader.
- Optionally a small router + switch for a separate network (GL.iNet routers work well, with VPN).
- A camera (often the phone) for documentation.

→Worth considering

- Power bank, spare SD card, noise-isolating headphones, USB drive with write-block option (Kanguru / Netac).

→Maintenance

None of this is worth much if it is not reviewed and updated. Keep control of the bag and its documents — make a checklist of what belongs in it so it is easy to maintain.

> A **locked-off room** is a must: it removes the analyst from prying eyes and reduces a real stress factor. Remember colleagues or suspects are innocent until proven otherwise.

Action Cards

A set of action cards for incident-response situations — short, descriptive playbooks for the field. Not exhaustive; adapt them as you like. All under Creative Commons 4.0.

CC 4.0 PLAYBOOKS

→Incident-response action cards

The author's contribution of action cards for incident-response situations — not exhaustive, and you are encouraged to modify them. Credit for the original work is appreciated, and the author would love to hear how you adapt them. All PDFs are under Creative Commons 4.0.

PDF **Live Data Acquisition**
151 KB · CC 4.0

PDF **Malware Analysis**
169 KB · CC 4.0

PDF **Forensics Analysis**
171 KB · CC 4.0

PDF **Acquisition**
162 KB · CC 4.0

PDF **Replication**
165 KB · CC 4.0

DFIR Links

A curated collection of DFIR links gathered over time — frameworks, memory and malware references, data-collection tools and sample sources.

REFERENCES CURATED

→Curated DFIR links

A collection of references gathered over time, grouped by topic.

→General / mixed

- https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf
- https://github.com/Neo23x0
- https://github.com/google/grr
- https://github.com/meirwah/awesome-incident-response
- http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html
- https://github.com/gchq/CyberChef

→Memory forensics

- https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples
- https://digital-forensics.sans.org/media/volatility-memory-forensics-cheat-sheet.pdf

→Malware analysis

- https://github.com/rshipp/awesome-malware-analysis
- https://github.com/Yara-Rules/rules
- https://remnux.org/
- https://thedfirreport.com/2020/10/18/ryuk-in-5-hours/
- https://evasions.checkpoint.com/

→Data collection & appliances

- https://github.com/orlikoski/CyLR/releases
- https://securityonionsolutions.com/
- https://cybersecurity.att.com/products/ossim
- https://wazuh.com/

→Network & convert

- https://www.netresec.com/?page=PcapFiles
- https://gchq.github.io/CyberChef/
- https://stegtool.net/

DEFENCIA

Part II · Forensics & Evidence

Forensics & Analysis

The fundamentals of forensic data handling — write protection, storage and exchange — plus an introduction to Autopsy, the open-source forensic framework built on The Sleuth Kit.

AUTOPSY | SLEUTH KIT | FREE

→Write protection, storage & exchange

Write protection lets you read but not write a medium, so no timestamps are made and the evidence is not contaminated. If you genuinely cannot write-block, fall back to a normal connection — but document exactly what was done, ideally with witnesses or even on film. Store data securely (who has access? is the room logged?) and have a defined way to exchange images with partners and authorities.

→Autopsy — the forensic framework

Autopsy is a forensic framework built on The Sleuth Kit (TSK) from the Linux world, now used worldwide. It loads image files — commonly E01 (EnCase) and DD (raw) — and can carve data, extract web cache and visited sites, read registry hives, hash and compare against known files, and much more.

> Autopsy works right after installation — just tell it where to save output (the indexed data can be several GB per case, so ensure free space).

→Ingest modules

Ingest modules are Autopsy's built-in automation — they look for specific data such as databases, GPS data, carved (deleted) files and search history. Several are pre-installed; more can be downloaded from the community modules repo and Python plugins dropped into the autopsy `python_module` folder.

> Autopsy requires training and knowing what you are looking for. Commercial tools ship ready-made search filters for scenarios — but they cost a lot of money.

CHAIN OF CUSTODY

EVIDENCE INTEGRITY · COC · LEGAL

Chain of Custody

Correct evidence securing requires the right workflow. The most important thing is that the integrity of effects and data can be demonstrated at every step.

LEGAL PROCESS

→Why it matters

Correct evidence handling requires the right workflow. The key is that the integrity of effects and data can be demonstrated throughout the process — so if it reaches court, there is no doubt about what was done.

> Doubt is always raised in a lawsuit — that is the defence's main job. The more thorough you have been, the less the prosecutor's case can be doubted.

→Data collection & securing workflow

When collecting effects for examination: document, document, document.

- Time and location.
- Who was present (in personnel matters, that HR was present too).
- How many effects were collected; what data, and how.
- How data was secured — with or without write protection, and why.
- Who is responsible for the effects until stored in the forensic room or safe.
- A list of hand-overs between colleagues — **this is the heart of CoC**.

→When evidence changes hands

Each hand-over carries a signature and a description of the effects involved — it can be largely automated with a form.

→During transport

For important cases, effects must be transported so their physical integrity can be demonstrated — proving nothing was tampered with. Tamper-evident bags can be bought (e.g. Miladan, Scenesafe).

DATA COLLECTION

ACQUISITION · INTEGRITY · DOCUMENTATION

Data Collection

Proper data collection is one of the most important things in an incident. Authorities, third parties and courts all depend on a controlled, well-documented process.

ACQUISITION INTEGRITY

→Why proper data securing matters

Data collection is one of the critical things in an attack. Authorities and third parties need to understand what happened from your collection. If the case goes to court and the process was not controlled, the case may fall.

→What data do we have?

What data is available if things go wrong? You can often retrieve a computer and secure it — but what about a cloud service? Do you know how to retrieve that data, and how long the export takes? Test this before you get hit.

→Questions before collecting

Describe your process — the more careful, the better the collection.

- Date/time of start and end.
- Performed by whom?
- What data is retrieved?
- Which tools (and versions) were used?
- Data from AV/IDS/IPS/network gear — which software version?
- How were integrity checks made, and with what tool?
- What does the data contain — format, fields (a description of the data)?
- Where is data stored and how is access granted?

→Integrity

Create a process for collecting data, store it on a solution you control, then run a tool like DirHash to calculate hash values of the collected files. Calculation time roughly mirrors copy time — it is bound by disk I/O and CPU.

Hashing

A hash is a cryptographic fingerprint of a file — like DNA. In forensics it lets you prove integrity and identify whether two files are identical.

INTEGRITY OPEN TOOLS

→What is a hash value?

A way of cryptographically calculating the value of a file — a bit like DNA for humans. Hash algorithms have existed for many years and are used for many things, including masking stored passwords.

→How we use it in forensics

Hashing produces a unique value for a file, so you can identify whether two files are the same. You see this when downloading an ISO (e.g. Kali Linux) and checking the SUM — calculate the hash yourself and compare it to the vendor's published value. In Autopsy you can hash files and compare against the NIST NSRL database.

→Tools

A myriad of free / open-source tools exist.

Tool	Use
HashTools	Calculate hash for one or many files; red line on mismatch.
DirHash	Calculate hashes for folders + subfolders.
Autopsy / NSRL	Hash files and compare against the NIST National Software Reference Library.

```
DirHash.exe C:\Users\<username>\Downloads\folder -sum -t output_folder -progress
```

DirHash example — hash a folder tree with progress output.

Memory Forensics

Memory forensics means looking for artifacts in RAM — password hashes, process lists, network connections and more — primarily using Volatility 3.

VOLATILITY 3 CASES

→What is memory forensics?

Looking for artifacts in memory — not always easy, but valuable. Useful artifacts include password hashes, network cache, the process list (proof of execution), command history, DLL lists and network connections (netscan).

→Volatility 3

There is a downloadable Volatility 3 cheat-sheet on the presentations page. Plugin families exist for Windows, Linux (linux.pslist, linux.bash, linux.malfind…) and Mac (mac.pslist, mac.netstat…).

Volatility 3 uses OS-based plugins instead of the version-specific profiles of 2.x — it works across a wider range of dumps and is a bit faster. Clone it and run with Python 3 (note: REMnux does not ship Volatility 3 by default).

```
git clone https://github.com/volatilityfoundation/volatility3
python3 vol.py -h        # list functions
python3 vol.py -f dump.mem windows.pslist > pslist.txt
```

Pipe output to a file, then grep it later — easier searching and you keep the output.

→Worked case — pull a password

Secure a memory dump with FTK Imager (or suspend a VM and take the .vmem). Then run the hashdump plugin to recover a machine hash.

```
vol.py -f KeepassMemdump.mem windows.hashdump.Hashdump
# output: 43239E3A0AF748020D5B426A4977D7E5
```

Example: extracting a stored hash from a Windows memory image.

→KeePass vulnerability example

An example of validating a real KeePass 2.5.3 vulnerability with keepass-password-dumper: install KeePass 2.5.3, create a vault, dump the process via Task Manager, install .NET 7 and run the dumper.

```
dotnet run C:\Path\To\Dump\KeePass.DMP
```

The recovered master password is largely reconstructed — only the first character is missing.

> This sensitive topic is shown to illustrate why patching and memory hygiene matter — not to enable misuse.

USB for Live Forensics

A USB for live forensics needs the right, tested toolset. Use quality media, prepare it properly, and cover both *nix and Windows environments.

ACQUISITION WIP

→Build a live-forensics USB

Many programs are worth having on a USB for live forensics. Test what works for you and your environment first. Note there are fewer tools for *nix systems — the landscape skews Windows — but plenty of Linux/Unix servers still exist, so account for them.

→Prepare the USB

Make sure the USB is completely erased and of good quality.

- Buy from a known brand — Verbatim, SanDisk, Kingston. A cheap unknown USB is not good enough.
- Always use a brand-new one straight out of the box.
- Consider keeping a stack of factory-new USBs in the jump-bag, plus one write-block-capable USB (e.g. Netac) for storing your toolset read-only.

→Two scenarios

- **Lite version** — lightweight, for specific purposes; carry tools for both *nix and Windows environments.
- Categories to cover: acquisition programs, memory dump, and a live-boot option.

Autopsy Plugins

A small collection of custom Autopsy Python plugins that extend the framework toward your investigations — each with a SHA-256 for integrity.

AUTOPSY PYTHON

→Custom Autopsy plugins

A small collection of the author's own Autopsy Python plugins. Each is distributed as a ZIP with a SHA-256 for integrity. Drop the Python module into Autopsy's `python_module` folder.

→URLcheck (URLhaus)

Checks URLs against URLhaus. Version 1.11 avoids accumulation when run multiple times and simplifies copy-paste to a text file.

ZIP **URLcheck v1.11**
4.04 KB · SHA-256 673d98...83d

→Pi-hole lookup

Similar to URLcheck, but uses your own Pi-hole as the reference to flag sites worth digging into. Version 1.0 — still under development and test.

ZIP **Pi-hole lookup v1.0**
3.06 KB · SHA-256 b4755f...b61

→MalwareIndicator

An experimental plugin (v1.0 / detection logic v4.2) for identifying malware behaviours using a generic approach.

ZIP **MalwareIndicator v1.0**
4.02 KB · SHA-256 1fe877...257

DEFENCIA

Part III · Malware & Labs

MALWARE ANALYSIS

REMNUX · STATIC · DYNAMIC

Malware Analysis

Analysing an unknown sample — how it spreads, what it creates and what it contacts — gives the organisation a head start on blocking it. A REMnux-based static and dynamic workflow.

REMNUX STATIC + DYNAMIC

→REMnux workflow

Inspired by Lenny Zeltser's talks. Download REMnux, update it (first run takes a while), then transfer the sample over SFTP (start sshd, find IP with ifconfig). Extract password-protected archives with p7zip.

```
sudo apt-get update
sftp://<ip-address>     # user: remnux
7z x filename.zip       # prompts for password
```

Get the sample onto REMnux and unpack it.

→Static analysis

Look at API calls, strings and structure without running the file.

Command	What it shows
yara-rules <file>	Match against YARA rules.
clamscan <file>	AV scan (update first with `freshclam`).
manalyze <file>	PE analysis — imports, compile date, language hints.
peframe <file>	PE overview; can feed YARA.
pecheck <file>	Entropy (hints at packing/compression/payload); look for overlay.
strings / pestr <file>	Readable strings — and whether the two differ.

→Dynamic analysis

Detonate the sample in an isolated VM and observe behaviour — new files and processes, network domains/IPs contacted. Always do this in a disposable, network-isolated environment.

> Combine with the **Kasm** sandbox or a dedicated malware-lab VM on a separate network segment — never on your daily machine.

ASSEMBLYLINE · DOCKER · SELF-HOST

Malware Lab — Assemblyline

A getting-started guide for standing up Assemblyline — the Canadian open-source malware-analysis platform — on a self-hosted Docker server.

ASSEMBLYLINE DOCKER

→Assemblyline

A getting-started guide for the CSE/Canada open-source malware-analysis platform. Tested on a server with 32 GB RAM, 16 cores, 360 GB disk (Hetzner CX53) running updated Ubuntu 24.04 with the latest Docker.

→Install Docker

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
# add repo, then:
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo docker run hello-world
```

Standard Docker CE install from the official repo.

→Install Assemblyline

This is a getting-started guide, not comprehensive. Always set strong passwords and run it on an isolated host.

```
git clone https://github.com/CybercentreCanada/assemblyline-docker-compose.git ~/deployments/assemblyline
cd ~/deployments/assemblyline
# edit .env — set strong passwords, choose SERVICE_VERSION=4.6.stable, minimum or full
```

Clone the compose deployment and adjust the .env file before bringing it up.

Labs for Analysis

How to prepare your labs — space, hardware and software requirements — aimed at reusing older hardware rather than buying the latest and greatest.

HARDWARE VMS

→Preparing labs

Have labs ready for in-depth analysis. The author aims to use older, hand-me-down hardware rather than the latest and greatest — that is often the reality.

→Forensic lab

A forensic lab needs its own machine — analysing and indexing data is heavy. The constraints are disk read/write speed, RAM and CPU.

- RAM: 16 GB minimum; forensic tools build an index/database of artifacts and consume a lot.
- CPU: analysis spikes to 80–100% — more cores help considerably.
- Disk: the single most important factor. An NVMe PCIe drive can be ~20× faster than an old 7200 RPM SATA disk, saving many hours.

→Example hardware

Component	Spec
RAM	128 GB
Disks	1× 1 TB NVMe (OS) · 2× 2 TB (image analysis + index) · 1× 8 TB (storage/temp)
CPU	Intel i9 Extreme or Ryzen Threadripper
GPU	RTX 3080 / AMD 6900 XT — must support CUDA

> Can less do it? Yes — an older i7, 16 GB RAM and a SATA SSD will still let you investigate; it just takes longer. Start there for the first year or two and build knowledge.

→Virtualization & ready-made VMs

For both new and old PCs, start with VMware or VirtualBox on a Linux host (Ubuntu / Linux Mint) — Linux leaves more resources for the VMs.

- **Security Onion** — a complete SIEM (Zeek), analysis, capture and IR setup as an ISO.
- **REMnux** — Linux toolkit for malware analysis.
- **Kali / Ubuntu** — general-purpose offensive and base systems.

LOKI

IOC SCANNER · YARA · HASHES

Loki

Loki is a simple IOC scanner: point it at files or folders and it matches them against known hashes and YARA rules you create or pull from the repository.

YARA IOC

→Loki — IOC scanner

Loki is a tool for detecting YARA and hash values you have discovered while handling an incident. Run it from the command line to scan files or folders against known hashes and YARA rules — either your own, or pulled from the repository.

> Source: github.com/Neo23x0/Loki

DEFENCIA

Part IV · Governance & Law

PRIVACY · PRINCIPLES · DATA SUBJECT RIGHTS

GDPR

A compressed walkthrough of GDPR: the data controller's obligations, the seven data-protection principles, data subject rights, and how breaches are handled.

GOVERNANCE ISO 27701

→Obligations of the data controller

The controller decides why and how personal data is processed and is responsible for compliance. GDPR has applied since 25 May 2018 and is built on seven core principles.

→The 7 data protection principles

Principle	Meaning
Transparency & lawfulness	Processing must be lawful, fair and transparent to the data subject.
Purpose limitation	Collect for specified, explicit, legitimate purposes only.
Data minimisation	Collect only what is adequate, relevant and necessary.
Accuracy	Keep data correct and up to date; erase or rectify errors.
Storage limitation	Keep data no longer than necessary for the purpose.
Confidentiality & integrity	Protect against unauthorised access, loss or damage (security).
Accountability	Be able to demonstrate compliance with all of the above.

→Data subject rights

- Duty to provide information & right of access.
- Right to rectification and the right to be forgotten (erasure).
- Right to restrict processing and data portability.
- Right to object; protection against solely automated decisions and profiling.

→Data breach through GDPR's eyes

A breach involving personal data must be handled under GDPR. Map your data (data mapping) so you know what you hold and where, and align handling with ISO 27701 (the privacy extension to ISO 27001).

> On personal-data loss, involve the DPO and keep contact with the supervisory authority — in Denmark, Datatilsynet — within the notification deadline.

Paragraphs — Danish Law

Some of the Danish criminal-law sections that apply to cyber attacks — a quick reference, not legal advice.

REFERENCE DENMARK

→Danish criminal law on cyber

Some of the Danish criminal-law sections aimed at cyber attacks. (Reference only — not legal advice.)

§	Offence
§ 263a	Distribution of means of unauthorised access.
§ 264	Unauthorised access.
§ 235	Child pornography.
§ 279	Fraud.
§ 279a	Data fraud.
§ 281	Extortion.
§ 288	Robbery.
§ 290a	Money laundering.
§ 291	Vandalism.
§ 293	Theft (DDoS via § 293 stk. 2).
§ 193	Public disturbance.
§ 299b	Copyright.
§ 268	Slander (serious untrue accusation, or spread via mass media).

References: danskelove.dk/straffeloven · danskelove.dk/nis-loven · anklagemyndigheden.dk knowledge base.

TEMPLATES

GOVERNANCE · PDCA · DOCUMENTS

Templates

Governance document templates to use as a starting point — built around the PDCA cycle and Continual Service Improvement. Adapt them to your business.

GOVERNANCE CC / FREE

→Governance templates

Examples developed from the author's knowledge, internet inspiration and AI (then adjusted). Use, modify and adapt them at your own responsibility — a starting point, not a source of truth.

> Familiarise yourself with the **PDCA cycle** (Plan, Do, Check, Act). You don't start with the perfect document — you start somewhere and refine it over time (Continual Service Improvement).

→Downloads

PDF **BIA Template**
Business Impact Assessment · 180 KB

PDF **CCP Template**
Crisis Communication Plan · 201 KB

PDF **NDA Template**
Non-Disclosure Agreement · 165 KB

PDF **Vendor Selection Questionnaire**
198 KB

PDF **AUP Template**
Acceptable Use Policy · 169 KB

PDF **SDD Template**
Solution Design Document · 243 KB

XLSX **Risk Analysis Register**
13 KB

Intelligence in Business

One of the easiest and cheapest things a business can do is monitor the media for intelligence. A few hours a week watching security news via RSS adds real value.

OSINT RSS

→Building intelligence cheaply

There are many ways to build intelligence in a company — it depends on need and investment, and you need skilled staff. But there is a cheap entrance that adds real value: spending a few hours a week watching headlines, news stories and attack vectors in the surrounding world.

→What is RSS?

RSS (Really Simple Syndication) is an update feed — co-founded by Aaron Swartz. You subscribe to a site's feed and stay current whenever new material is published, watching many sites at once. You just need an RSS reader.

→Trustworthiness

Prefer getting news directly from the source rather than having others interpret it. A good reader lets you check whether a story appears across several media you have chosen — adding validity. Keep your critical sense; distinguishing good sources from bad has only gotten harder.

→RSS readers

The author wants a reader that is online and supports keywords for watching specific topics.

- Inoreader
- Feedly
- NewsBlur
- The Old Reader

> See also the self-hosted Miniflux guide and the news-screening workflow for an automated, self-hosted version of this.

Abbreviations

A glossary of the DFIR and governance abbreviations that come up most often — handy when the acronyms start flying.

GLOSSARY

→DFIR

Term	Meaning
Image / Image file	A raw copy of original hardware (HDD, SSD, SD, USB).
Actioncard	A short, descriptive way of approaching a task, aligned with management expectations.
CTF	Capture The Flag — a challenge to find hidden treasures.
CoC	Chain of Custody — documenting when and how evidence changes hands.
DD	Data Duplication — an uncompressed image file format.
DFIR	Digital Forensics and Incident Response.
E01	EnCase image file (raw drive data); compressed or uncompressed.
Forensics	A scientific way of finding the truth of what happened.
Live Image Boot	Bootable drive (e.g. CAINE, Paladin) for acquisition where hardware can't be removed.
Live triage	Triage and acquisition from a running system.
Playbook	A guided way of approaching a problem (see also actioncards).
Triage / SWB / WB	Quick pre-analysis · Software Write-Blocker · Write-Blocker.

→Governance

Term	Meaning
BCP	Business Continuity Plan (a.k.a. contingency / beredskabsplan).
CB	Certification Body — responsible for e.g. ISO certification.
CMMC	Cybersecurity Maturity Model Certification.
FUD	Fear, Uncertainty and Doubt.
GRC	Governance, Risk Management and Compliance.
Risk appetite	The level of risk an organisation is willing to accept.
PCI DSS	Payment Card Industry Data Security Standards.
TNO / ZT	Trust No One · Zero Trust — strict access control and encryption around your data.

DEFENCIA

Part V · Foundations & Learning

Linux

If you have never used Linux, getting started takes a little work — but it is free, light on resources, and a multitool for DFIR once you learn the terminal.

FREE / OSS | DEBIAN-BASED

→Intro to Linux

If you have not used Linux, getting started takes a little extra work — but there is plenty of help online and most things are easier than they look. Linux is free and open source, needs fewer resources than Windows, and can breathe life into an old PC.

→Packages

Linux ships many distributions differing in look and installed software (called **packages**). Package management differs per distro — RPM (Red Hat Package Manager) and DPKG (Debian) are the two main families. The course uses Debian-based Linux (Ubuntu, Linux Mint, Debian, REMnux).

→The terminal

The terminal — the Command Line Interface (CLI) — is where Linux shows its power, and for many it is a learning curve: new commands, flags and ways of working in a text box.

```
lablo@linuxserver$ sudo apt-get update
```

A typical terminal prompt and an update command.

Windows CMD

A collection of built-in Windows commands — useful for looking into a system's network, configuration and integrity without installing any tools.

REFERENCE BUILT-IN

→Built-in Windows commands

An old collection of built-in Windows commands — sometimes useful for inspecting a system without installing anything.

Command	Does
ipconfig	IP configuration for all adapters.
netstat	All active network connections.
nbtstat	NetBIOS-over-TCP/IP stats and connections.
nslookup	Returns your local DNS server.
ping	Sends data to a host/IP.
hostname	Returns the computer's name.
msconfig	Edit startup configuration.
msinfo32	System information utility.
regedit	Registry editor (remote: connect network registry).
sfc /scannow	System File Checker.
chkdisk	Repair damaged files.
cleanmgr	Disk cleanup.
MMC	Microsoft Management Console.

SOFTWARE

TOOLSET · LICENSING · DFIR

Software

An overview of the software covered in the course — imaging, forensics, virtualization, SIEM and documentation tools — with a note on licensing for each.

TOOLSET MOSTLY FREE

→Software used in the field

An overview of the tools covered, with licensing. Replacements happen, so this can change.

Tool	Purpose	License
Autopsy	Forensic framework (Sleuth Kit); ingest modules + plugins.	Free / OSS
VMware / VirtualBox	Virtualization. VMware is the author's favourite; VirtualBox is free.	~$200 / Free
EmEditor	Text editor that opens files up to 250 GB — handy for huge logs on Windows.	$259 lifetime / $40 yr
Splunk	SIEM; quick to install and start analysing logs. Free up to 500 MB/day.	Free tier
SOF-ELK	SANS (Phil Hagen) ELK appliance; recognises common log formats, but needs Elastic/Kibana/Logstash knowledge to customise.	Free
Linux	A multitool in itself — does a lot with few resources.	Free
SimpleMind Pro	Mind mapping — don't underestimate it for summarising topics.	~$28 lifetime
Screenpresso	Image / video / audio capture for documentation.	Free / ~$45
Windows	OS; trial license works 90 days with full functionality.	OEM / Volume
FTK Imager	Free imaging tool (AccessData) — AD1, DD, E01. USB media may need a separate write-blocker.	Free
dd / dc3dd / dcfldd	Linux imaging tools; variants add hashing and progress bars.	Free / OSS

Backup

Personal opinions on backup, built on a few basic principles. Start with 3-2-1, then choose a cloud provider — and add your own encryption for sensitive data.

3-2-1 CLOUD

→The 3-2-1 principle

A few basic principles cover most people and SMBs (and large firms too): keep **3** copies of your data, on **2** different media, with **1** copy off-site. Christopher Barnatt's "Explaining Computers" has a good video on it.

→Cloud recommendations

Service	Notes
Jottacloud	Scandinavian (Norway). App backs up files live; versioning allows rollback (useful against ransomware). ~690 DKK/yr unlimited for one user. Holds the keys, so could in theory read your files — mitigate with client-side encryption.
pCloud	Switzerland; creates a virtual drive so you access all files directly. Optional client-side **Crypto** folder Pcloud cannot read; share links with passwords and expiry dates.
Boxcryptor	A service for encrypting files before they reach any cloud — so the provider cannot read them.

→On encryption & trust

"Am I safe?" is always debatable. For backup and restore these services are stable and versioned. Whether a provider reads your files is hard to disprove when they hold the keys — so for sensitive data, add your own client-side encryption.

> This page reflects personal opinion and is not sponsored.

Course & Peripherals

You should be able to start with forensics and attack management without spending the whole budget. Here are the minimum hardware, peripherals and software to get going.

REQUIREMENTS BUDGET

→Getting started without breaking the budget

Everyone should be able to start with forensics and incident management without spending the whole budget — at minimum a proof of concept. You can learn software on small, retired computers and scale up later as your needs become clear.

→Minimum PC requirements

Work on a stand-alone machine — the course handles malware that can infect a system even with precautions.

- i5 processor (or equivalent).
- 16 GB RAM.
- 250 GB+ storage, ideally SSD (forensic software wants a fast disk).
- A few USB sticks.
- Able to run virtualization — the course runs two VMs alongside the host OS.

→Peripherals & software

Have at least one 16 GB+ USB (ideally USB 3.0), for data collection and a bootable CAINE / Paladin Linux stick. Course software is open source and free; some examples use VMware (e.g. for the network setup).

- FTK Imager (free, requires registration) — for acquisition.
- CAINE / Paladin — bootable, write-protected Linux for collection.
- VMware / VirtualBox — virtualization.

LITERATURE

BOOKS · COURSE MATERIAL · E-BOOK

Literature

The literature used in the DFIR and Governance courses — plus the entire site compiled into a downloadable e-book.

BOOKS E-BOOK

→Course literature

The literature used in the SIR/DFIR and Governance courses at KEA (subject to change).

- **Incident Response & Computer Forensics**, 3rd ed. (ISBN 9780071798686) — goes hand in hand with how the author was trained; many elements worth their weight in gold.
- **IT Governance** and a dedicated **GDPR** book — a good introduction to the field, written by long-standing experts.
- A Linux book is recommended if you have not worked in Linux before.

→Download this site as an e-book

A compiled offline copy of the website.

PDF **DFIR Knowledge — Stop Bad Things**
14.45 MB · use at your own risk

PRACTICE · CASES · SELF-STUDY

Sites for Learning

Curiosity leads to growth. A collected list of websites with practice cases and challenges to help you find your own learning path.

SELF-STUDY FREE + PAID

→Challenge your unknowns

Learning new things sparks curiosity. The author's advice to students asking "what's next?" is twofold: be curious and expose yourself to an area you don't know (failing and restarting is how you learn), and play with assignments and cases you find on the web.

PDF **Links for Experts**
261 KB · collected learning sites

> Some content is free, some paid. Set aside time and prepare — it will be time-consuming and at times frustrating. Don't worry, it will come.

DOCUMENTATION · CAPTURE · MIND MAPS

Software for Study & Productivity

Creating good material is half the work. The author's favourite tools for documentation and learning — screen capture and mind mapping.

DOCUMENTATION FREE + PAID

→Screenshots & screencasts

The author's go-to is **Screenpresso** — one of the cheapest, easiest tools for screenshots and short screencasts, always at hand. The paid version (~240 DKK/yr) adds OCR text extraction, branding removal, blurring of sensitive data, HD video capture and annotation.

→Mind mapping

Mind mapping is an easy way to get an overview of a lot of knowledge. The author used the free, open-source **FreeMind** for years, then switched to the paid **SimpleMind** for its ease of use and steady improvement.

- Intuitive, easy to use.
- Add and reorganise topics quickly.
- Free-form and auto layout.

SANTAFUN MINI CTF 2023

CHALLENGE · STEGANOGRAPHY · MEMORY

Santafun Mini CTF 2023

A festive memory-forensics and steganography challenge: a leprechaun crew has ransomed the beer list inside a released image. Can you decode it?

CTF FUN

→Santafun mini CTF 2023

A festive memory-forensics / steganography challenge. A leprechaun group "Lebr3c0rnz4mash" has ransomed the author's beer list, hidden inside a released image — and hints at widely-used web encodings and hex tampering. Decode the image to recover the list.

PNG **santas-little-helper.png**
8.23 MB · the released challenge image

A fun way to practise carving, decoding and memory artifacts. Grab the image and see what you can recover.

ABOUT

DEFENCIA · THE AUTHOR · MISSION

About

Defencia = Defense + CIA (Confidentiality, Integrity, Availability). A place to share DFIR and governance knowledge from teaching at KEA.

ABOUT

→What is Defencia?

Defencia is a combination of **Defense** and **CIA** — Confidentiality, Integrity, Availability.

→Why this page?

A place to share assignments, knowledge and teaching material from KEA — based on the author's own experience, with inspiration sought from the best in the industry. Teaching is a spare-time passion: knowledge should be shared, because that is how we best prevent and protect against attacks.

> If it inspires even one or two people to avoid or handle an attack — at work or privately — then it's mission accomplished.

→Who is behind it?

A random guy from Denmark with an interest in information and IT security, and the motivation to share knowledge.

DEFENCIA

Part VI · Tool Guides & Workflows

AUTOPSY

DFIR · DISK IMAGE · DEAD-BOX FORENSICS

Autopsy

Graphical forensics platform built on top of The Sleuth Kit. Analyse disk images, build timelines, search keywords, carve deleted files and run automated artifact modules — in a case-oriented interface.

OPEN SOURCE | WINDOWS · LINUX | BASIS TECHNOLOGY

TSK	E01/DD	Ingest
engine under the hood	image formats	module pipeline

→What it is

Autopsy is a case-based GUI for dead-box forensics: you add a disk image as a data source, run ingest modules that automatically extract artifacts, and analyse the results in a timeline and tree view. The engine is The Sleuth Kit (TSK) — the command-line tools can also be run directly.

> **Platform reality:** Autopsy is primarily developed for **Windows** and is clearly most stable there. The Linux version can be built but needs more setup (manual TSK + Java dependencies). For serious casework the Windows build is recommended — for example in an isolated analysis VM or Kasm workspace.

→Windows installation

recommended

Download the MSI from autopsy.com. The installer bundles TSK, the Java runtime and Solr text indexing.

Step

```
# 1. Get the MSI from autopsy.com/download
# 2. Verify SHA-256 against the site
# 3. Run installer (bundles TSK + JRE + Solr)
# 4. Launch → New Case → enter case name + examiner
```

Allocate plenty of RAM/disk — Solr indexing of large images is heavy.

> **Best practice:** Always work on a **copy** of the image, never the original. Use a write-blocker during acquisition and document hashes (MD5+SHA256) before and after. Autopsy verifies the image hash automatically when you add it.

→Linux installation

advanced

Requires The Sleuth Kit, Java and building from source. Expect more troubleshooting than on Windows.

Dependencies (Ubuntu/Zorin)

```
sudo apt update
sudo apt install sleuthkit openjdk-17-jdk -y
```

Autopsy

```
# get the ZIP from github.com/sleuthkit/autopsy/releases
unzip autopsy-*.zip && cd autopsy-*
bash unix_setup.sh
./bin/autopsy
```

> **Alternative on Linux:** If the Autopsy GUI gives you trouble, use the Sleuth Kit CLI directly — same engine, fully scriptable, and often faster for targeted tasks.

→Case workflow

The typical sequence from image to report.

Step	What happens
New Case	Create a case with name, number and examiner — all work is isolated per case
Add Data Source	Add a disk image (E01/DD/VMDK), local disk or logical file folder
Configure Ingest	Choose modules to run automatically during import
Analyze	Review the tree: file types, deleted files, web history, EXIF, keyword hits
Timeline	Visualise events chronologically (MAC times, web, log events)
Tag & Report	Tag findings, export a report (HTML/Excel/KML)

→Ingest modules

Modules that automatically extract artifacts on import. The most important ones:

Module	Function
Hash Lookup	Match files against known hash sets (NSRL, custom whitelist/blacklist)
Keyword Search	Index text and search words/regex/patterns (cards, email, URL)
File Type ID	Identify file types by signature (not extension)
Extension Mismatch	Flag files where the extension does not match the actual content
Recent Activity	Web history, USB devices, installed software, executed programs
EXIF Parser	Extract metadata/GPS from images
PhotoRec Carver	Carve deleted files from unallocated space
Email Parser	Parse PST/MBOX/EML mailboxes
Android / iOS Analyzer	Mobile artifacts (requires add-ons)
YARA	Run YARA rules against file content during ingest

→The Sleuth Kit CLI

scriptable

Same engine as Autopsy, but on the command line — ideal on Linux and in automation.

Command	Function
`mmls image.dd`	Show partition table / layout
`fsstat -o OFFSET image.dd`	File-system details for a partition
`fls -r -o OFFSET image.dd`	List files/folders recursively (incl. deleted)
`icat -o OFFSET image.dd INODE`	Extract file content by inode number
`istat -o OFFSET image.dd INODE`	Metadata for a specific inode
`blkcat / blkls`	Read / extract raw data blocks (unallocated)
`mactime -b body.txt`	Build a timeline from an fls/ils bodyfile
`tsk_recover -o OFFSET image.dd OUT/`	Bulk-recover all files to a folder
`tsk_gettimes image.dd`	Generate a bodyfile for mactime directly

> **Timeline on the CLI:** `tsk_gettimes image.dd > bodyfile && mactime -b bodyfile -d > timeline.csv` — produces a sortable CSV without opening the GUI.

VELOCIRAPTOR

DFIR · ENDPOINT VISIBILITY · THREAT HUNTING

Velociraptor

Open-source endpoint monitoring and digital forensics. Collect artifacts, hunt threats and run live response across thousands of machines via VQL — Velociraptor Query Language.

OPEN SOURCE | LINUX · WINDOWS · MACOS | RAPID7 / COMMUNITY

1	VQL	~300+
binary, no deps	query language	built-in artifacts

→What it is

A single static binary that is both server and client. The server has a web GUI; clients (agents) roll out on endpoints and call home over TLS. Everything is driven by VQL, so you can ask arbitrary questions of endpoints in real time instead of waiting for a fixed feature set.

> **For your test environment:** Velociraptor can also run entirely without a server — as a standalone "offline collector" that packages a single executable you run on a suspect machine. See Offline collector.

→Server installation

Linux

Get the latest binary from the GitHub releases. The interactive config generator creates server and client configuration plus the first admin login.

Download binary

```
# check github.com/Velocidex/velociraptor/releases for the latest version
wget https://github.com/Velocidex/velociraptor/releases/latest/download/velociraptor-linux-amd64
chmod +x velociraptor-linux-amd64
sudo mv velociraptor-linux-amd64 /usr/local/bin/velociraptor
```

Generate configuration (interactive)

```
velociraptor config generate -i
```

*Asks about SSL mode (self-signed / Let's Encrypt / autocert), ports and data path. Produces **server.config.yaml** + **client.config.yaml**.*

Create admin user

```
velociraptor --config server.config.yaml user add admin --role administrator
```

Install as systemd service

```
velociraptor --config server.config.yaml service install
sudo systemctl enable --now velociraptor_server
```

*The GUI now runs at **https://your-server:8889** (default).*

> **Behind Nginx:** Since you already run Nginx + Certbot on Hetzner, you can set Velociraptor to listen on localhost and reverse-proxy the GUI at `velo.defencia.dk` with your existing cert. Remember the websocket-upgrade headers in the proxy block.

→Deploy clients

The client uses the same binary with `client.config.yaml`. Package an MSI/DEB or run it directly.

Linux client (direct)

```
sudo velociraptor --config client.config.yaml client -v
```

Build Debian package

```
velociraptor --config client.config.yaml debian client
```

Build Windows MSI

```
velociraptor.exe --config client.config.yaml msi
```

Verify client connection

```
# in the GUI: Search → show active clients
# or via API/notebook with VQL: clients()
```

→VQL — basics

query language

VQL resembles SQL but plugs into live system data. Run queries in the GUI's Notebook or via the CLI `velociraptor query`.

Run VQL from CLI

```
velociraptor --config server.config.yaml query \
  "SELECT Name, Pid, Ppid FROM pslist()"
```

Find listening network connections

```
SELECT * FROM netstat()
WHERE Status = 'LISTEN'
```

Search files with the glob plugin

```
SELECT FullPath, Mtime FROM glob(
  globs="/home/**/*.sh")
WHERE Mtime > now() - 86400
```

VQL plugin	Function
`pslist()`	Running processes with metadata
`netstat()`	Network connections
`glob()`	File search with wildcard patterns
`hash()`	Compute MD5/SHA1/SHA256 of files
`yara()`	Run YARA rules against files or process memory
`pe_dump() / authenticode()`	Inspect PE files and signatures
`parse_evtx()`	Parse Windows event logs
`parse_mft()`	Parse the NTFS Master File Table
`execve()`	Run an external command and capture output
`artifact_set() / Artifact.*`	Call reusable artifact definitions

→Hunts & artifacts

A "hunt" runs an artifact collection across all (or a subset of) clients at once. Artifacts are reusable VQL packages — 300+ ship with it.

> **Typical DFIR flow:** Pick an artifact (e.g. `Windows.Detection.Yara.Process` or `Linux.Sys.BashShell`) → start a hunt against a label group → results are collected centrally → export to CSV/JSON for further analysis. You can upload your own YARA rules (signature-base) as a parameter to the yara artifacts.

List all artifacts

```
velociraptor artifacts list
```

Show an artifact's definition

```
velociraptor artifacts show Windows.Detection.Yara.Process
```

→Offline collector

standalone

Package a self-contained executable that collects artifacts without a server — perfect for a single suspect machine or where you cannot deploy agents.

Build offline collector (interactive)

```
velociraptor --config server.config.yaml collector
# choose artifacts → produces Collector_*.exe / .bin
# run on the target → output is a zip with all artifacts
```

→Hardening

> **Important:** The Velociraptor server is effectively a C2-like capability — full command execution on all clients. Restrict GUI access behind VPN/Nginx with an IP whitelist, use strong roles (`reader` vs `investigator` vs `administrator`), and rotate client certificates if an endpoint is compromised. Log all hunts.

> **For your stack:** Run the server on a dedicated VLAN/segment, put the GUI behind `velo.defencia.dk` with a Fail2ban jail on the login endpoint (see Fail2ban — coming as its own guide), and use UFW to allow the client port only from your endpoint subnet.

KAPE

Kroll Artifact Parser and Extractor. Collects and parses the most relevant forensic artifacts from a live or mounted Windows system in minutes — built for fast triage when time is critical.

FREE — NOT OSS | WINDOWS ONLY | KROLL · ERIC ZIMMERMAN

Targets what gets collected **Modules** how it gets parsed **gkape** GUI · **kape** CLI

→What it is

KAPE has two phases. **Targets** define which artifacts are copied (registry hives, event logs, prefetch, browser history, etc.) — even locked system files, via raw disk access. **Modules** then run parsing tools (many of Eric Zimmerman's EZ Tools) against the collected data and produce readable CSV/JSON.

> **License honesty:** KAPE is **free to use** but **not open source** — it is governed by Kroll's EULA and requires registration to download. Strictly speaking it does not belong in an "open source" catalogue, but it is included because it is central to modern DFIR triage. The Targets/Modules definitions (.tkape/.mkape), by contrast, are community-maintained on GitHub under MIT.

→Setup

Windows

Step

```
# 1. Download from kroll.com/kape (requires registration)
# 2. Extract the ZIP — KAPE is portable, no installation
# 3. Run Get-KAPEUpdate.ps1 for the latest Targets/Modules
# 4. gkape.exe = GUI · kape.exe = command line
```

Run from a USB/external drive to minimise the footprint on the target system.

Update definitions

```
.\Get-KAPEUpdate.ps1
```

→Targets — collection

Targets (.tkape) are YAML-like definitions of which files/paths to copy. Compound targets combine several into one.

Target	Collects
!SANS_Triage	Compound — broad triage set recommended as the default
!BasicCollection	Core artifacts: registry, logs, prefetch, $MFT
RegistryHives	SYSTEM, SOFTWARE, SAM, SECURITY, NTUSER.dat
EventLogs	Windows .evtx event logs
Prefetch	.pf files (program-execution evidence)
FileSystem	$MFT, $LogFile, $UsnJrnl, $J
WebBrowsers	Chrome/Edge/Firefox history, cache, cookies
LNKFilesAndJumpLists	Shortcut and jumplist artifacts

→Modules — parsing

Modules (.mkape) run external tools against the collected artifacts. Many wrap Eric Zimmerman's EZ Tools.

Module / tool	Function
`!EZParser`	Compound — runs the whole EZ Tools suite against targets
`PECmd`	Parse Prefetch → program-execution timeline
`MFTECmd`	Parse $MFT / $J → file timeline
`EvtxECmd`	Parse event logs → normalised CSV
`RECmd`	Registry parsing with batch plugins
`AmcacheParser`	Amcache.hve → installed/executed software
`LECmd / JLECmd`	LNK and JumpList parsing
`SBECmd`	ShellBags → folder-browsing history

After parsing: Many open the CSV output in **Timeline Explorer** (also EZ Tools) or load it into Elastic/Splunk. Combine with Autopsy for deeper disk analysis of the same image.

→CLI examples

Triage live C: with the SANS target

```
kape.exe --tsource C: --target !SANS_Triage \
  --tdest E:\out\%m --gui
```

Collect + parse in one go

```
kape.exe --tsource C: --target !SANS_Triage \
  --tdest E:\out \
  --module !EZParser --mdest E:\parsed
```

__%m__ = machine name · __%d__ = timestamp in the destination path

Flag	Function
`--tsource`	Source (drive letter, image or folder)
`--target`	Target(s) to collect
`--tdest`	Destination for collected artifacts
`--module`	Module(s) to run for parsing
`--mdest`	Destination for parsed output
`--vhdx / --zip`	Package the collection as a VHDX container or ZIP
`--vss`	Include Volume Shadow Copies
`--debug / --trace`	Detailed logging

→Triage flow

Classic IR sequence: KAPE collects artifacts in minutes (Targets) → parses them to CSV (Modules/!EZParser) → the analyst triages in Timeline Explorer → suspicious machines get deep disk analysis in Autopsy or live response via Velociraptor. KAPE is the "quick overview", the others are the "deep dive".

CLAMAV & YARA

DFIR · MALWARE SCANNING · DETECTION RULES

ClamAV & YARA command reference

A comprehensive overview of the functions, commands and flags for both tools — tailored to Linux (Zorin / Ubuntu).
Click a code example to copy it.

9	60+	2	v1.x / v4.x
core commands	flags & options	integration methods	ClamAV / YARA

01 Installation

Packages from the Ubuntu/Zorin repo. `clamav-daemon` provides background scanning; `clamtk` is a GUI.

Install ClamAV + YARA + GUI apt

```
sudo apt update
sudo apt install clamav clamav-daemon clamtk yara -y
```

*Packages: **clamav** (clamscan/freshclam), **clamav-daemon** (clamd/clamdscan), **clamtk** (GUI), **yara** (standalone engine).*

02 freshclam update

Fetches and updates virus definitions (`main.cvd`, `daily.cvd`, `bytecode.cvd`). Normally runs as a service in the background.

Manual update

```
# Stop the service first – otherwise it locks the database
sudo systemctl stop clamav-freshclam
sudo freshclam
sudo systemctl start clamav-freshclam
```

Flag	Function
`-v`	Verbose output during download
`--daemon` / `-d`	Run as a background daemon with periodic checks
`--checks=N` / `-c N`	Number of database checks per day (daemon mode)
`--datadir=PATH`	Path to the database folder (default `/var/lib/clamav`)
`--config-file=PATH`	Use an alternative `freshclam.conf`
`--show-progress`	Show a progress bar during download
`--on-update-execute=CMD`	Run a command after a successful update
`--list-mirrors`	Show the status of known mirrors

03 clamscan on-demand scanner

Standalone scanner. Loads the whole signature database into RAM on each run — thorough but slow. Use clamdscan for frequent scanning.

Scan folder, show detections only

```
clamscan -ri ~/Downloads
```

***-r** recursive · **-i** infected only*

Scan with beep on detection

```
clamscan -ri --bell ~
```

Move detections to quarantine

```
mkdir -p ~/quarantine
clamscan -ri --move=~/quarantine ~
```

*Prefer **--move** over **--remove***

Scan the whole system

```
sudo clamscan -ri / --exclude-dir="^/sys|^/proc"
```

Flag	Function
`-r` / `--recursive`	Scan subfolders recursively
`-i` / `--infected`	Show infected files only
`-o`	Suppress OK results (errors/detections only)
`--bell`	Audible signal on detection
`--remove[=yes/no]`	Delete infected files (destructive — be careful)
`--move=DIR`	Move detections to a quarantine folder
`--copy=DIR`	Copy detections to a folder (keep the original)
`-l FILE` / `--log=FILE`	Write the scan result to a log file
`-d FILE/DIR` / `--database=`	Use your own signature database or YARA rule
`--exclude=REGEX`	Skip files by regex on the path
`--exclude-dir=REGEX`	Skip folders by regex
`--include=REGEX`	Scan only matching files
`--max-filesize=N`	Skip files larger than N (e.g. 100M)
`--max-scansize=N`	Max data scanned per file
`--max-recursion=N`	Max depth in archive/compression layers
`--scan-archive[=yes/no]`	Scan inside zip/rar/tar etc. (default yes)
`--detect-pua[=yes]`	Detect Potentially Unwanted Applications
`--scan-pe / --scan-elf / --scan-ole2`	Toggle scanning of specific file types
`--alert-encrypted[=yes]`	Flag encrypted archives/documents as suspicious
`--alert-broken[=yes]`	Flag broken executables
`--alert-macros[=yes]`	Warn about macros in Office documents
`--bytecode[=yes]`	Enable bytecode signatures (advanced detection)
`--gen-json`	Generate JSON metadata about scanned objects
`-z` / `--allmatch`	Continue scanning a file after the first match
`--stdout`	Send all output to stdout (errors too)
`-V` / `--version`	Show version + database info

Exit codes: `0` = no detections · `1` = malware found · `2` = error. Useful in scripts: `clamscan -ri ~ || echo "found!"`

04clamdscan

daemon client

Sends scan jobs to the running `clamd` daemon, which already has the database in memory. Markedly faster for repeated scans.

Fast scan via daemon

```
clamdscan -r ~/Downloads
```

Multithreaded file passing

```
clamdscan --multiscan --fdpass ~
```

Flag	Function
`-m` / `--multiscan`	Scan in parallel with multiple threads
`--fdpass`	Pass the file descriptor to the daemon (avoids permission issues)
`--stream`	Stream file content to the daemon over a socket
`-i` / `--infected`	Show detections only
`--move=DIR` / `--remove`	Quarantine / deletion as in clamscan
`--config-file=PATH`	Alternative `clamd.conf`
`-l FILE`	Log file

05clamd & service

daemon

The background daemon that keeps the database in RAM and offers on-access scanning (real-time monitoring of the filesystem via fanotify).

Enable & start daemon

```
sudo systemctl enable --now clamav-daemon
```

On-access (real-time) scan

```
# requires OnAccess* in clamd.conf
sudo clamonacc -v
```

clamd.conf option	Function
`OnAccessIncludePath`	Folder monitored in real time
`OnAccessPrevention`	Block access to infected files (not just alert)
`OnAccessExtraScanning`	Also scan on create/move events
`MaxThreads`	Number of scan threads in the daemon
`LocalSocket`	Unix socket path for client communication
`LogFile` / `LogTime`	Log destination and timestamping

> **clamconf** dumps the entire active configuration for troubleshooting: `clamconf -n`

06sigtool

signatures

A tool for inspecting databases and building your own signatures — useful when you create custom detection in DFIR cases.

Command	Function
`sigtool --info FILE.cvd`	Show metadata about a database file (version, signature count)
`sigtool --unpack FILE.cvd`	Unpack the database into raw signature files
`sigtool --md5 FILE`	Generate an MD5 hash signature of a file
`sigtool --sha256 FILE`	Generate a SHA256 hash signature
`sigtool --hex-dump`	Convert input to hex for use in signatures
`sigtool --find-sigs=REGEX`	Find signatures matching a pattern in the database
`sigtool --vba FILE`	Extract VBA macros from Office documents

Your own hash signature: `sigtool --sha256 malware.bin > mine.hsb` → put the `.hsb` in `/var/lib/clamav/` and scan with `clamscan -d mine.hsb`

07yara

detection rules

Pattern-based classification of files and processes from rules you write yourself. Standalone YARA supports all modules (pe, elf, hash, math, cuckoo and more) — unlike ClamAV's built-in YARA support.

Run rule against a folder

```
yara -r rules.yar ~/case
```

Show matching strings

```
yara -r -s rules.yar ~/case
```

Compile rules (faster)

```
yarac rules.yar rules.cmp
yara -r rules.cmp ~/case
```

Scan a running process

```
sudo yara rules.yar $(pidof suspect_process)
```

YARA can scan PIDs directly — powerful for memory forensics

Ready-made rule sets: Neo23x0/signature-base (Florian Roth, the engine behind LOKI/THOR), Elastic protections-artifacts, Yara-Rules/rules. Clone and run recursively: `yara -r signature-base/yara/ ~/case 2>/dev/null`

08 YARA flag-reference

A complete flag overview for the `yara` CLI.

Flag	Function
`-r` / `--recursive`	Scan folders recursively
`-s` / `--print-strings`	Show the strings that matched
`-m` / `--print-meta`	Show metadata from the rule (author, ref, desc)
`-g` / `--print-tags`	Show rule tags
`-e` / `--print-namespace`	Show the namespace of a matched rule
`-L` / `--print-stats`	Show statistics after scanning
`-c` / `--count`	Show only the number of matches per rule
`-d VAR=VALUE`	Define an external variable for the rule
`-t TAG` / `--tag=TAG`	Show only rules with this tag
`-i ID` / `--identifier=`	Run only the rule with this name
`-n` / `--negate`	Show files that do **not** match
`-w` / `--no-warnings`	Suppress warnings
`-f` / `--fast-scan`	Fast scan (stop at the first match per string)
`-x MODULE=FILE`	Pass module data (e.g. cuckoo report) to the rule
`-p N` / `--threads=N`	Number of threads for parallel scanning
`-a N` / `--timeout=N`	Abort scanning after N seconds
`-z N` / `--max-strings-per-rule`	Cap on strings per rule
`--max-process-memory-chunk`	Chunk size for process scanning
`-X N` / `--skip-larger=N`	Skip files larger than N bytes
`--scan-list`	Treat input as a file with a list of paths
`-v` / `--version`	Show the YARA version

Anatomy of a rule: `rule name { meta: … strings: $a = "x" condition: $a }` — the **meta**, **strings** and **condition** sections. Modules are imported at the top with `import "pe"`.

09 YARA in ClamAV

integration

ClamAV can load YARA rules natively alongside its own signatures.

Add YARA rules to ClamAV's database

```
sudo cp rules.yar /var/lib/clamav/
# or scan ad hoc with -d:
clamscan -r -d rules.yar ~/case
```

Limitations: ClamAV's YARA engine does **not** support all features. Modules like `pe`, `hash`, `math` and external variables do not work, and rules using them are rejected on load. For full functionality — including signature-base — use standalone YARA.

10 Automation

systemd / cron

Schedule weekly scanning and logging.

Weekly cron scan with log

```
# crontab -e  → run every Sunday at 02:00
0 2 * * 0 clamdscan -r --fdpass /home \
  -l /var/log/clamav/weekly-$(date +\%F).log -i
```

Combined ClamAV + YARA wrapper

```
#!/usr/bin/env bash
TARGET="${1:-$HOME}"
TS=$(date +%F_%H%M)
clamdscan -r --fdpass -i "$TARGET" | tee "clam_$TS.log"
yara -r -m ~/yara/signature-base/yara/ "$TARGET" \
  2>/dev/null | tee "yara_$TS.log"
```

KASM WORKSPACES

SANDBOX · BROWSER ISOLATION · DISPOSABLE DESKTOPS

Kasm Workspaces

Stream containerised desktops and apps straight into your browser. Each session is a disposable environment reset on logout — ideal for opening suspicious links, detonating malware or analysing phishing without touching your own machine.

COMMUNITY EDITION (OSS) | DOCKER / LINUX | KASM TECHNOLOGIES

Browser all access via the web

Disposable reset per session

Docker container per workspace

→What it is — and what it is not

Kasm is **not** a forensics tool. It is a platform that runs desktops and applications in Docker containers and streams the image to your browser via a web server. Keyboard and mouse go in, pixels come out — the code never runs on your local machine.

Why it belongs in a DFIR catalogue: It gives you **disposable burner environments**. When you need to investigate a phishing link from `phishing@defencia.dk`, open a suspicious document, or detonate a sample — you do it in a Kasm session that is thrown away afterwards. Combined with network isolation it becomes a safe detonation chamber.

> **Important caveat:** Kasm isolates via containers, not a full VM/hypervisor. For **real** malware detonation (ransomware, a sample attempting container escape) you should place Kasm inside an isolated VM on a separate network segment — do not rely on the container boundary alone.

→Installation

Docker

The Community Edition has a single installation script that sets up all services in Docker (database, agent, proxy, GUI).

Requirements

```
# Recommended: a dedicated Ubuntu VM
# Min. ~4 vCPU / 8 GB RAM / 50 GB disk for multiple sessions
# Docker is installed by the script if missing
```

Download & install Community Edition

```
# check kasmweb.com/downloads for the latest version
cd /tmp
curl -O https://kasm-static-content.s3.amazonaws.com/kasm_release_VERSION.tar.gz
tar -xf kasm_release_*.tar.gz
sudo bash kasm_release/install.sh
```

*The script prints autogenerated passwords for **admin** and **user** at the end — save them.*

Access

```
# GUI at https://<server-ip>  (default port 443)
# log in as admin@kasm.local
```

Behind your Nginx: As with your other services, Kasm can sit behind `lab.defencia.dk` with your Certbot cert. Note that Kasm terminates TLS itself — consider a stream proxy or adjust Kasm's own cert config rather than double TLS termination.

→Workspaces & images

A "workspace" is a Docker image with a desktop or app. Kasm maintains a registry of ready-made images.

Workspace type	Use
Kasm Desktop (Ubuntu)	Full Linux desktop for general analysis
Chrome / Firefox	Isolated browser — open suspicious links safely
Tor Browser	Anonymous OSINT / investigation without leaking your own IP
Remnux (custom)	Build your own image with a malware-analysis toolset
Kali (custom)	Offensive/testing tools in a disposable environment

Custom images: Kasm images are ordinary Docker images built on their base. You can create a custom workspace with, say, Remnux or your own analysis tools pre-installed, push it to your own registry and add it in the admin panel.

→DFIR / phishing workflow

Concrete usage that fits your Defencia setup:

Scenario	Approach
Phishing-link	Start a disposable Chrome session → open the URL → observe the redirect chain, landing page, credential harvest — with no risk to your own machine
Mistænkt dokument	Upload to a Linux desktop session → open in the isolated environment → inspect macros/payloads → throw the session away
OSINT på trusselsaktør	Tor browser session → investigate without revealing your infrastructure or IP
Sample-detonering	Custom Remnux image in a network-isolated session → observe behaviour (only in a fully isolated VM, see below)

Integration: What you collect (URLs, IOCs, hashes) feeds straight into your n8n phishing-analyzer workflow and your Defencia Intel Dashboard.

→Isolation & networking

Network segmentation is essential: Put the Kasm VM on a dedicated, isolated VLAN (like your Wu_IOT pattern) with no access to your internal network. For detonation: no route to production, only controlled egress (optionally via an analysis proxy so you capture C2 traffic). Assume everything run in a session is hostile.

Control	Recommendation
Network	Isolated VLAN, no route to internal services
Egress	Controlled/logged egress via an analysis proxy
Session-levetid	Set a short timeout and force destroy on logout
GUI-adgang	Behind VPN + Fail2ban on login (see Fail2ban — coming)
Host	Kasm in its own VM, not on a host with other services

→Operations

Service status

```
sudo docker ps --filter name=kasm
```

Stop / start the whole stack

```
sudo /opt/kasm/bin/stop
sudo /opt/kasm/bin/start
```

Update images

```
# in the admin GUI: Workspaces → select image → Update
# or pull a new image and point the workspace at the new tag
```

Logs

```
sudo docker logs kasm_api
tail -f /opt/kasm/current/log/*.log
```

DOCKER

INFRASTRUCTURE · CONTAINERS · SELF-HOSTING

Docker

Container runtime and the foundation under nearly every self-hosted service. Package an application with all its dependencies into an isolated, reproducible image and run it identically on any Linux host.

ENGINE: OPEN SOURCE | LINUX | DOCKER INC.

Image	Container	Compose
immutable template	running instance	multi-container stacks

→Concepts

Three core words you meet constantly: an **image** is an immutable template (code + dependencies), a **container** is a running instance of an image, and **Compose** describes one or more containers declaratively in a YAML file so the whole stack starts with one command.

> **For your stack:** Both n8n and Miniflux are most easily deployed as Docker Compose stacks — this guide is the foundation for both.

→Installation

Ubuntu / Debian

Use Docker's official apt repo rather than the distro package — it is newer and includes Compose v2 as a plugin.

1 · Remove old packages & add prerequisites

```
sudo apt remove docker docker-engine docker.io containerd runc # if present
sudo apt update
sudo apt install ca-certificates curl gnupg -y
```

2 · Add Docker's GPG key & repo

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
  sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo $VERSION_CODENAME) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

*On Zorin/Debian derivatives: if `$VERSION_CODENAME` is not recognised by the repo, set it manually to the Ubuntu/Debian base your version is built on (e.g. **noble** or **bookworm**).*

3 · Install Engine + Compose plugin

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin -y
```

4 · Verify

```
sudo docker run --rm hello-world
docker compose version
```

→Post-install

Run Docker without sudo and let the daemon start at boot.

Add your user to the docker group

```
sudo usermod -aG docker $USER
# log out/in or:
newgrp docker
```

Enable at boot

```
sudo systemctl enable --now docker
```

> **Security note:** Membership of the `docker` group is effectively root access (you can mount the host filesystem into a container). Grant it only to trusted users, and consider rootless mode on multi-user systems.

→Docker Compose

deklarativt

Describe your stack in `compose.yaml` and manage it with `docker compose`. Example skeleton:

compose.yaml (minimal)

```yaml
services:
  app:
    image: nginx:latest
    restart: unless-stopped
    ports:
      - "127.0.0.1:8080:80"
    volumes:
      - ./data:/usr/share/nginx/html:ro
    networks: [ web ]

networks:
  web:
```

*Bind to **127.0.0.1** so the port is local only and exposed solely via your Nginx reverse proxy — not directly to the internet.*

Command	Function
`docker compose up -d`	Start the stack in the background (detached)
`docker compose down`	Stop and remove containers + network
`docker compose pull`	Pull the latest images
`docker compose logs -f`	Follow logs live
`docker compose ps`	Status of the stack's services
`docker compose restart`	Restart services
`docker compose exec app sh`	Open a shell in a running container

Update routine: `docker compose pull && docker compose up -d` pulls new images and recreates only the containers that changed.

→CLI reference

Command	Function
`docker ps` / `docker ps -a`	Running / all containers
`docker images`	List local images
`docker logs -f NAME`	Follow a container's logs
`docker exec -it NAME sh`	Interactive shell in a container
`docker stop / start / restart NAME`	Lifecycle control
`docker rm NAME` / `docker rmi IMAGE`	Remove container / image
`docker inspect NAME`	Full JSON metadata (network, mounts, config)
`docker stats`	Live CPU/RAM/IO per container
`docker system df`	Disk usage split across images/containers/volumes
`docker system prune -a`	Clean unused images/containers/networks (free space)

Careful with `prune -a`: It removes all images not in use by a container — including ones you want to keep. Add `--volumes` only if you deliberately want to delete unmounted volumes (can mean data loss).

→Networking

Containers on the same user-defined network can reach each other by service name as hostname — that is how n8n finds its database.

Type	Use
bridge (default)	Default isolated network per Compose stack; service name = DNS
host	Shares the host's network stack directly (no isolation — avoid if possible)
none	No networking — full isolation
internal	Bridge with no outbound internet — good for databases

Pattern: Put the database on an `internal` network so it cannot reach the internet, and the application on both the internal and a proxy network. Never expose the database port on the host.

→Volumes & persistence

Containers are ephemeral — data that must survive a recreation has to go in a volume or bind-mount.

Named volume (Docker-managed)

```
docker volume create n8n_data
docker volume ls
docker volume inspect n8n_data
```

Bind-mount (host path)

```
# in compose: ./data:/app/data
# gives you direct access to the files on the host
```

Backup: Named volumes live under `/var/lib/docker/volumes/`. For your restic/pCloud stack you can either back up that path, or better: `docker compose exec db pg_dump …` for consistent database dumps rather than copying raw files while the database is running.

→Hardening

Control	Recommendation
Port-binding	Bind to `127.0.0.1`, expose only via reverse proxy
restart-policy	`unless-stopped` so services come up after a reboot
Image-tags	Pin to specific versions, not blindly `latest`, in production
read-only	`read_only: true` + `tmpfs` where possible
capabilities	`cap_drop: [ALL]` and add back only what is needed
no-new-privileges	`security_opt: [no-new-privileges:true]`
UFW	Note: Docker manipulates iptables and can bypass UFW rules — bind to localhost rather than relying on UFW alone
Updating	Regular `pull` + recreate; monitor base-image CVEs

The Docker + UFW trap: Docker writes its own iptables rules and can publish ports past your UFW rules. If you bind a port to `0.0.0.0`, it is often open to the internet regardless of UFW. The fix: always bind to `127.0.0.1:PORT` for internal services, or use the `ufw-docker` project to fix the rule ordering.

N8N (DOCKER)

AUTOMATION · WORKFLOWS · SELF-HOSTING

n8n · Docker

Self-hosted workflow automation with a visual node editor. Connect APIs, databases and services into pipelines without hosting it at a third party. This guide runs n8n as a Docker Compose stack with PostgreSQL behind it.

FAIR-CODE LICENSE | DOCKER / LINUX | N8N GMBH

PostgreSQL **Webhooks** **Queue**

recommended backendrequire a public URLmode for scaling

→Before you start

n8n can run with an internal SQLite file, but for anything serious you should use **PostgreSQL** — it is more robust with many workflows and concurrent runs. Prerequisite: a working Docker installation.

> **Prerequisite:** Follow the Docker guide first. For webhooks (e.g. your phishing analyzer or RSS triggers) n8n needs a public URL via a reverse proxy — see Reverse proxy.

→Compose stack

PostgreSQL

A complete stack with n8n + Postgres on an isolated network. Put it in a folder, e.g. ~/n8n/.

compose.yaml

```yaml
services:
  postgres:
    image: postgres:16-alpine
    restart: unless-stopped
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    volumes:
      - pg_data:/var/lib/postgresql/data
    networks: [ internal ]
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -U $${POSTGRES_USER}" ]
      interval: 10s
      retries: 5

  n8n:
    image: docker.n8n.io/n8nio/n8n:latest
    restart: unless-stopped
    depends_on:
      postgres:
        condition: service_healthy
    environment:
      DB_TYPE: postgresdb
      DB_POSTGRESDB_HOST: postgres
      DB_POSTGRESDB_DATABASE: ${POSTGRES_DB}
      DB_POSTGRESDB_USER: ${POSTGRES_USER}
      DB_POSTGRESDB_PASSWORD: ${POSTGRES_PASSWORD}
      N8N_HOST: ${N8N_HOST}
      N8N_PORT: 5678
      N8N_PROTOCOL: https
      WEBHOOK_URL: https://${N8N_HOST}/
      GENERIC_TIMEZONE: Europe/Copenhagen
      N8N_ENCRYPTION_KEY: ${N8N_ENCRYPTION_KEY}
    ports:
      - "127.0.0.1:5678:5678"
    volumes:
      - n8n_data:/home/node/.n8n
    networks: [ internal, web ]

volumes:
  pg_data:
  n8n_data:

networks:
  internal:
    internal: true
  web:
```

*Postgres sits on **internal** (no internet), n8n on both internal and web. The port is bound to **127.0.0.1** — reachable only via the reverse proxy.*

→Configuration (.env)

Put secrets in a `.env` file next to compose.yaml — never in the YAML itself.

.env

```
POSTGRES_USER=n8n
POSTGRES_PASSWORD=# generate: openssl rand -base64 24
POSTGRES_DB=n8n
N8N_HOST=n8n.defencia.dk
N8N_ENCRYPTION_KEY=# generate: openssl rand -hex 32
```

Critical — encryption key: `N8N_ENCRYPTION_KEY` is used to encrypt all stored credentials. If it is not set explicitly, it is autogenerated in the volume — and **if you lose the volume without having the key, all credentials are irrecoverably lost.** Store the key in Bitwarden alongside your other secrets.

Key env variable	Function
`N8N_ENCRYPTION_KEY`	Encrypts stored credentials (save it!)
`WEBHOOK_URL`	Public base URL n8n builds webhook addresses from
`N8N_HOST / N8N_PROTOCOL`	Hostname and protocol behind the proxy
`GENERIC_TIMEZONE`	Timezone for cron/schedule nodes
`N8N_SECURE_COOKIE`	Set `true` behind HTTPS (default)
`EXECUTIONS_DATA_PRUNE`	Auto-clean old execution logs
`N8N_RUNNERS_ENABLED`	Enable task runners (isolated code execution)

→Reverse proxy (Nginx)

n8n uses websockets for the live editor — the proxy block must handle upgrade headers.

/etc/nginx/sites-available/n8n.defencia.dk

```
server {
    listen 443 ssl;
    server_name n8n.defencia.dk;

    # ssl_certificate ... (Certbot inserts)

    location / {
        proxy_pass http://127.0.0.1:5678;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 3600s;
    }
}
```

*The long **read_timeout** prevents long-running workflows from being cut off by the proxy.*

Cert: `sudo certbot --nginx -d n8n.defencia.dk` — eller brug dit eksisterende wildcard for *.defencia.dk.

→Scaling — queue mode

advanced

For many concurrent or heavy workflows, n8n can run in **queue mode**: a main instance handles the UI/webhooks, and separate worker containers run jobs via Redis.

Components: Add a `redis` service, set `EXECUTIONS_MODE=queue` and `QUEUE_BULL_REDIS_HOST=redis` on both main and workers, and start n workers with the command `n8n worker`. Each worker pulls jobs from the queue — scale by increasing the number of worker containers.

When: For your current use (RSS screening, reports, generators) single-instance is fine. Queue mode only becomes relevant if you hit execution bottlenecks or want to isolate heavy AI calls from UI responsiveness.

→Operations

Start & follow logs

```
docker compose up -d
docker compose logs -f n8n
```

Update to latest

```
docker compose pull
docker compose up -d
```

Always read the release notes — n8n can have breaking changes between majors.

Backup database

```
docker compose exec postgres \
  pg_dump -U n8n n8n > n8n_$(date +%F).sql
```

Export workflows to file

```
docker compose exec n8n \
  n8n export:workflow --all --output=/home/node/.n8n/backup.json
```

> **For your backup stack:** Take both a `pg_dump` (data + credentials, encrypted with the encryption key) and a workflow export (portable JSON). Send both to restic/pCloud. Remember: without the encryption key, the credentials in the dump are useless.

→Hardening

Control	Recommendation
Exposure	Bind to `127.0.0.1`, only via Nginx + HTTPS
Database	Postgres on an `internal` network, no host port
Encryption key	Set explicitly, store in Bitwarden, back up separately from the volume
Login	Enable user management/2FA; Fail2ban jail on the login endpoint
Webhooks	Use webhook auth/tokens; validate payloads in the workflow
Task runners	Enable to isolate Code-node execution
Execution data	Prune old logs so they do not pile up sensitive data
Access	Consider VPN-only access to the editor; webhooks separate

> **Remember the Docker+UFW trap:** Bind n8n to `127.0.0.1:5678` — not `0.0.0.0` — otherwise the port may be exposed to the internet past UFW. See the Docker guide for details.

OPENCVE (DOCKER)

VULNERABILITY MANAGEMENT · CVE · SELF-HOSTING

OpenCVE · Docker

Self-hosted platform to monitor CVEs and get notified when vulnerabilities hit the products, vendors and technologies you subscribe to. Acts as the trigger source for your enrichment pipeline via webhooks.

OPEN SOURCE · BSD-3 | DOCKER / LINUX | OPENCVE.IO

Postgres Scheduler Webhook

+ Redis backendimports CVE data+ email notifications

→Architecture

OpenCVE consists of a web app, a database (PostgreSQL), a cache/queue (Redis) and a scheduler/worker that regularly fetches and indexes CVE data from the official sources. You subscribe to vendors/products, and the platform notifies you on changes.

> **In your pipeline:** OpenCVE is the *trigger* layer. When a relevant CVE appears, OpenCVE calls a webhook → your n8n workflow enriches it → the result lands in the CVE dashboard. See the CVE enrichment workflow. Prerequisite: Docker.

→Compose stack

official

OpenCVE provides an official Compose setup. The easiest path is to clone their repo and use the bundled `docker-compose.yml` + `.env`, since the scheduler/worker and web app share configuration.

1 · Get the official compose setup

```
# check github.com/opencve/opencve for the latest structure/version
git clone https://github.com/opencve/opencve.git
cd opencve
```

Versions and service names can change between releases — verify against their documentation before running.

2 · Configure .env

```
# copy the example and fill in secrets
cp .env.example .env
nano .env
# set POSTGRES_PASSWORD, SECRET_KEY (openssl rand -hex 32),
# mail settings and optionally START_URL / public hostname
```

3 · Start the stack

```
docker compose up -d
docker compose ps
```

Expected services: webapp, postgres, redis, scheduler and one or more workers.

> **Bind to localhost:** Make sure the webapp port is exposed only on `127.0.0.1` in compose and placed behind your Nginx reverse proxy — not directly to the internet. See the Docker+UFW trap in the Docker guide.

→Initialisation & CVE import

On first start the database must be migrated, a superuser created, and the initial CVE dataset imported. The first import is large and can take a while.

Migrate & create superuser

```
docker compose exec webapp \
  opencve upgrade
docker compose exec webapp \
  opencve create-user lars admin@defencia.dk --superuser
```

Command names can vary between versions — check `opencve --help` in the container.

Initial data import

```
docker compose exec webapp \
  opencve import-data
# fetches CVE/CWE/vendor data — takes time the first time
```

> **Scheduler:** After the first import, the scheduler keeps the database updated automatically on an interval. Check that the scheduler and worker containers are running and not logging errors.

→Subscriptions

The heart of OpenCVE: subscribe to the vendors and products relevant to your environment, so you only get noise from what matters.

Step	Action
Find vendor/produkt	Search Vendors/Products — e.g. Microsoft, Fortinet, Cisco, the products you actually run
Subscribe	Subscribe to a vendor (all products) or specific products
Organisations	Group subscriptions into org/project if you separate environments
Notifikation	Choose email and/or webhook per subscription/organisation

> **Curate tightly:** As with your RSS feeds, the value is in the curation. Subscribe only to what your organisation actually uses — otherwise the important CVEs drown in noise.

→Webhook notifications

to n8n

This is where OpenCVE connects to your automation. Create a webhook notification pointing at your n8n webhook URL, so every relevant CVE triggers the enrichment workflow.

Setup

```
# In the OpenCVE UI:
# Settings/Notifications → Add → Webhook
# URL: https://n8n.defencia.dk/webhook/opencve-cve
# Choose which subscriptions/severities trigger
# Optionally add a secret header for validation in n8n
```

OpenCVE POSTs a JSON payload with the CVE id, change type and metadata to your endpoint.

> **Next step:** n8n receives the payload and enriches the CVE (CVSS, EPSS, KEV, affected products) before it lands in the dashboard. That whole flow is described in the CVE enrichment workflow.

→Operations

Logs & status

```
docker compose ps
docker compose logs -f scheduler
docker compose logs --tail=50 webapp
```

Update

```
git pull
docker compose pull
docker compose up -d
docker compose exec webapp opencve upgrade
```

Always run migrations (upgrade) after an image update.

Backup database

```
docker compose exec postgres \
  pg_dump -U opencve opencve > opencve_$(date +%F).sql
```

Force a manual CVE sync

```
docker compose exec scheduler \
  opencve import-data # or the relevant sync command
```

→Hardening

Control	Recommendation
Exposure	Webapp bag Nginx + HTTPS, bind til `127.0.0.1`
Database/Redis	On an `internal` network, no host ports
SECRET_KEY	Strong, unique, stored in Bitwarden
Webhook	Secret header/token so only OpenCVE can trigger n8n
Login	Superuser with a strong password; Fail2ban on login
Mail	Use an app password/SMTP relay, not a personal password
Updating	Follow releases — vulnerability data needs a healthy scheduler

WORKFLOW

VULNERABILITY MANAGEMENT · N8N · OPENCVE → DASHBOARD

CVE enrichment

OpenCVE → dashboard

An n8n workflow that receives a webhook from OpenCVE when a subscribed CVE appears, enriches it with data from NVD and EPSS, and writes a combined record to your CVE dashboard database. A reference pattern with importable JSON.

N8N · REFERENCE TEMPLATE · OPENCVE · NVD · EPSS

7 **Webhook** **Upsert**

nodestrigger from OpenCVEto Postgres dashboard

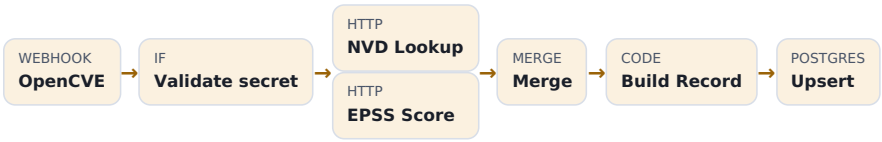
→What it does

OpenCVE tells you *that* a CVE is relevant, but a CVE id alone is not actionable. This workflow takes the id and gathers the context you need to prioritise: the CVSS score from NVD and the EPSS exploitation probability — and writes it all to your dashboard so you have one place to triage from.

Prerequisites: OpenCVE running with a webhook notification, n8n, and a Postgres table for the dashboard. This is a *reference pattern* — verify node types against your n8n version.

→Flow overview

Webhook validates → parallel enrichment → merge → build record → upsert.



Parallel enrichment: The NVD and EPSS lookups run simultaneously after secret validation and are combined in the Merge node (combine all). That keeps latency down compared with sequential calls.

→Node by node

#	Node	Type	Function
1	**Webhook**	`webhook`	POST endpoint `/webhook/opencve-cve` that OpenCVE calls
2	**Validate Secret**	`if`	Checks the secret header so only OpenCVE can trigger
3	**NVD Lookup**	`httpRequest`	Fetches CVSS + description from the NVD CVE API 2.0
4	**EPSS Score**	`httpRequest`	Fetches the EPSS probability from FIRST.org
5	**Merge**	`merge`	Combines the two enrichment sources (combine all)
6	**Build Record**	`code`	Assembles the fields into one flat dashboard object
7	**Upsert**	`postgres`	INSERT … ON CONFLICT updates existing CVE rows

NVD rate limits: The NVD API has tight rate limits without an API key (a few calls per 30 sec). Request a free NVD API key and add it as a header if you expect many CVEs — otherwise you risk 403/429 errors. Consider a small wait/retry node under load.

→Importable JSON

n8n import

Paste into n8n via **Workflows → ⋯ → Import from clipboard**. Replace `YOUR_SECRET`, the Postgres credential and the field paths to match your setup.

cve_enrichment_workflow.json

```
{
  "name": "OpenCVE Enrichment",
  "nodes": [
    {
      "parameters": {
        "httpMethod": "POST",
        "path": "opencve-cve",
        "responseMode": "lastNode",
        "options": {}
      },
      "name": "Webhook (OpenCVE)",
```

```
      "type": "n8n-nodes-base.webhook",
      "typeVersion": 2,
      "position": [
        240,
        300
      ]
    },
    {
      "parameters": {
        "conditions": {
          "options": {},
          "conditions": [
            {
              "leftValue": "={{ $json.headers['x-webhook-secret'] }}",
              "rightValue": "DIN_HEMMELIGHED",
              "operator": {
                "type": "string",
                "operation": "equals"
              }
            }
          ]
        }
      },
      "name": "Validate Secret",
      "type": "n8n-nodes-base.if",
      "typeVersion": 2,
      "position": [
        460,
        300
      ]
    },
    {
      "parameters": {
        "url": "=https://services.nvd.nist.gov/rest/json/cves/2.0?cveId={{ $json.body.cve_id }}",
        "options": {}
      },
      "name": "NVD Lookup",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.2,
      "position": [
        680,
        240
      ]
    },
    {
      "parameters": {
        "url": "=https://api.first.org/data/v1/epss?cve={{ $json.body.cve_id }}",
        "options": {}
      },
      "name": "EPSS Score",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.2,
      "position": [
        680,
        380
      ]
    },
    {
      "parameters": {
        "mode": "combine",
        "combineBy": "combineAll",
        "options": {}
      },
      "name": "Merge Enrichment",
      "type": "n8n-nodes-base.merge",
      "typeVersion": 3,
      "position": [
        900,
        300
      ]
    },
    {
      "parameters": {
        "jsCode": "// Saml beriget CVE-objekt til dashboard\nconst cve = $input.first().json;\nreturn [{ json: {\n  cve_id: cve.body?.cve_id ?? cve.cve_id,\n  cvss: cve.vulnerabilities?.[0]?.cve?.metrics?.cvssMetricV31?.[0]?.cvssData?.baseScore ?? null,\n  epss: cve.data?.[0]?.epss ?? null,\n  description: cve.vulnerabilities?.[0]?.cve?.descriptions?.find(d=>d.lang==='en')?.value ?? '',\n  published: cve.vulnerabilities?.[0]?.cve?.published ?? null,\n  enriched_at: new Date().toISOString()\n}}];"
      },
      "name": "Build Record",
      "type": "n8n-nodes-base.code",
      "typeVersion": 2,
      "position": [
        1120,
```

```
        300
      ]
    },
    {
      "parameters": {
        "operation": "executeQuery",
        "query": "INSERT INTO cves (cve_id, cvss, epss, description, published, enriched_at) VALUES ($1,$2,$3,$4,$5,$6) ON CONFLICT (cve_id) DO
UPDATE SET cvss=EXCLUDED.cvss, epss=EXCLUDED.epss, enriched_at=EXCLUDED.enriched_at;",
        "options": {}
      },
      "name": "Upsert Dashboard DB",
      "type": "n8n-nodes-base.postgres",
      "typeVersion": 2.5,
      "position": [
        1340,
        300
      ]
    }
  ],
  "connections": {
    "Webhook (OpenCVE)": {
      "main": [
        [
          {
            "node": "Validate Secret",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Validate Secret": {
      "main": [
        [
          {
            "node": "NVD Lookup",
            "type": "main",
            "index": 0
          },
          {
            "node": "EPSS Score",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "NVD Lookup": {
      "main": [
        [
          {
            "node": "Merge Enrichment",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "EPSS Score": {
      "main": [
        [
          {
            "node": "Merge Enrichment",
            "type": "main",
            "index": 1
          }
        ]
      ]
    },
    "Merge Enrichment": {
      "main": [
        [
          {
            "node": "Build Record",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Build Record": {
      "main": [
        [
```

```
          {
            "node": "Upsert Dashboard DB",
            "type": "main",
            "index": 0
          }
        ]
      ]
    }
  },
  "settings": {
    "executionOrder": "v1"
  }
}
```

After import — check: the webhook path matches the URL you set in OpenCVE, the secret header is real (not the placeholder), the Postgres credential is bound, and the JSON paths in Build Record match the actual API responses (NVD's structure is deeply nested and can change). That parser code is a starting point — test against a real response.

→Dashboard table

A minimal Postgres table that the Upsert node writes to. Adapt the fields to your dashboard.

schema.sql

```sql
CREATE TABLE cves (
  cve_id      TEXT PRIMARY KEY,
  cvss        NUMERIC,
  epss        NUMERIC,
  description TEXT,
  published   TIMESTAMPTZ,
  enriched_at TIMESTAMPTZ DEFAULT now()
);
```

If you use a SQLite-based CVE dashboard, swap the Postgres node for a SQLite/HTTP node but keep the same upsert logic (`INSERT … ON CONFLICT`).

→Credentials

Credential	Setup
Webhook secret	Any strong string; set in the OpenCVE webhook as a header and validated in node 2
NVD API key	Free from nvd.nist.gov; add as a header on the NVD node for a higher rate limit
Postgres	n8n Postgres credential against your dashboard database

→Tuning & extensions

Extension	Effect
CISA KEV	Add a lookup against CISA's Known Exploited Vulnerabilities — the strongest prioritisation signal
Severity filter	Drop/flag CVEs below a CVSS or EPSS threshold before upsert
Alerting	Add a Slack/email node on a KEV match or CVSS ≥ 9.0
Affected products	Store the CPE/product list from NVD so the dashboard can filter on your environment
Retry	Set retry-on-fail on the HTTP nodes against NVD/EPSS timeouts
Idempotens	`ON CONFLICT DO UPDATE` ensures repeated webhooks for the same CVE do not create duplicates

The whole chain: OpenCVE monitors and triggers → this workflow enriches → the dashboard triages. News/articles run separately via the Miniflux news workflow.

MINIFLUX (DOCKER)

RSS · FEED AGGREGATION · SELF-HOSTING

Miniflux · Docker

Minimalist, fast RSS reader written in Go — one binary, no JavaScript-heavy frontend, low resource use. Perfect as the feed engine behind automation via its REST and Fever APIs.

OPEN SOURCE · APACHE 2.0 | DOCKER / LINUX | FRÉDÉRIC GUILLOT

Go	PostgreSQL	REST + Fever
one small binary	the only backend	API for integration

→Why Miniflux

Deliberately minimalist: no social features, no bloated UI, extremely easy to run. It reads feeds, stores them in PostgreSQL and exposes a clean API — which makes it ideal as a data source for a threat-intelligence pipeline rather than just a reader.

> **For your pipeline:** Miniflux acts as the RSS engine that gathers your security feeds; n8n then pulls new articles via the API for screening/enrichment. Prerequisite: Docker installed.

→Compose stack

PostgreSQL

Miniflux **requires** PostgreSQL — there is no SQLite option. A stack with an isolated database and the admin created on first start.

compose.yaml

```yaml
services:
  db:
    image: postgres:16-alpine
    restart: unless-stopped
    environment:
      POSTGRES_USER: miniflux
      POSTGRES_PASSWORD: ${DB_PASSWORD}
      POSTGRES_DB: miniflux
    volumes:
      - mf_db:/var/lib/postgresql/data
    networks: [ internal ]
    healthcheck:
      test: [ "CMD", "pg_isready", "-U", "miniflux" ]
      interval: 10s
      retries: 5

  miniflux:
    image: miniflux/miniflux:latest
    restart: unless-stopped
    depends_on:
      db:
        condition: service_healthy
    environment:
      DATABASE_URL: postgres://miniflux:${DB_PASSWORD}@db/miniflux?sslmode=disable
      RUN_MIGRATIONS: 1
      CREATE_ADMIN: 1
      ADMIN_USERNAME: ${ADMIN_USERNAME}
      ADMIN_PASSWORD: ${ADMIN_PASSWORD}
      BASE_URL: https://rss.defencia.dk/
    ports:
      - "127.0.0.1:8080:8080"
    networks: [ internal, web ]

volumes:
  mf_db:

networks:
  internal:
    internal: true
  web:
```

RUN_MIGRATIONS *creates/updates the schema automatically.* ***CREATE_ADMIN*** *creates the admin user on first start from the env values.*

.env

```
DB_PASSWORD=# openssl rand -base64 24
ADMIN_USERNAME=lars
ADMIN_PASSWORD=# choose a strong password
```

→First start

Start the stack

```
docker compose up -d
docker compose logs -f miniflux
```

Create an extra admin manually (optional)

```
docker compose exec miniflux \
  miniflux -create-admin
```

> **Login:** Once the proxy is in place, go to `https://rss.defencia.dk` and log in with the admin user. Add feeds individually or import a whole **OPML** file under Settings → Import.

→Reverse proxy (Nginx)

/etc/nginx/sites-available/rss.defencia.dk

```
server {
    listen 443 ssl;
    server_name rss.defencia.dk;

    # ssl_certificate ... (Certbot inserts)

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

*Miniflux has no websockets — the proxy block is simpler than n8n's. **BASE_URL** in compose must match this hostname.*

> **Cert:** `sudo certbot --nginx -d rss.defencia.dk` eller dit wildcard for *.defencia.dk.

→API for automation

REST + Fever

This is where Miniflux shines as a pipeline component. Generate an API token under Settings → API Keys and call the REST API from n8n.

Endpoint	Function
`GET /v1/entries?status=unread`	Fetch unread articles (for screening in n8n)
`GET /v1/feeds`	List all subscribed feeds
`POST /v1/feeds`	Add a new feed programmatically
`PUT /v1/entries`	Mark articles read/unread in bulk
`GET /v1/entries?search=TERM`	Full-text search across feeds
`GET /v1/me`	Verify token / user info

Example — fetch unread via curl

```
curl -H "X-Auth-Token: YOUR_TOKEN" \
  "https://rss.defencia.dk/v1/entries?status=unread&limit=50"
```

> **Pipeline pattern:** n8n schedule trigger → call `/v1/entries?status=unread` → screen with Mistral → enrich → mark as read via `PUT /v1/entries`. The token is set as an n8n credential, so it never sits in cleartext in the workflow.

→Operations

Update

```
docker compose pull
docker compose up -d
```

*Migrations run automatically at startup due to **RUN_MIGRATIONS=1**.*

Backup database

```
docker compose exec db \
  pg_dump -U miniflux miniflux > mf_$(date +%F).sql
```

Export feeds (OPML)

```
# in the UI: Settings → Export → OPML
# or via API: GET /v1/export
```

Check status

```
docker compose ps
docker compose logs --tail=50 miniflux
```

> **For your backup stack:** Take an OPML export (your curated feeds, portable) plus a `pg_dump` (read/unread states, starred items). OPML alone is enough to rebuild the subscriptions if the database is lost.

→Hardening

Control	Recommendation
Exposure	Bind to `127.0.0.1:8080`, only via Nginx + HTTPS
Database	Postgres on an `internal` network, no host port
API token	Use a token, not basic auth, for automation; rotate as needed
Login	Strong admin password; Fail2ban jail on login
Feed proxy	Consider `FETCH_*` timeouts against slow/malicious feeds
Users	Create a separate non-admin user for the API if possible
Updating	Keep the image updated — the Go binary is small, upgrades are fast

> **Remember the Docker+UFW trap:** Bind to `127.0.0.1:8080` — never `0.0.0.0` for internal services. See the Docker guide.

WORKFLOW

THREAT INTEL · N8N · MINIFLUX → MISTRAL

News screening

from Miniflux

An n8n workflow that periodically fetches unread articles from Miniflux, lets a Mistral model decide relevance, enriches the relevant ones into short threat-intel summaries, and marks everything as read. A reference pattern with importable JSON — adapt it to your own feeds and nodes.

N8N | REFERENCE TEMPLATE | MINIFLUX API + MISTRAL

7 nodes **2-stage** screen → enrich **REST** Miniflux API

→What it does

Miniflux gathers your curated security feeds, but raw feeds are noisy. This workflow puts an intelligent filter on top: a cheap model screens for relevance, and only what passes goes on to more expensive enrichment. The result is a low-noise flow of relevant, summarised security news.

Prerequisites: Miniflux running with an API token, and n8n with Mistral credentials. This is a *reference pattern* — node types and parameters can differ from your n8n version, so verify against your editor.

→Flow overview

Two-stage screening: a cheap model filters out, an expensive model enriches only the relevant.

Branch at "Relevant?": true → enrich with Mistral Large → mark read. **false** → mark read directly (skipped). Both branches end by closing the article so it is not screened again.

→Node by node

#	Node	Type	Function
1	**Schedule Trigger**	`scheduleTrigger`	Runs every hour (adjust the interval to feed volume)
2	**Get Unread**	`httpRequest`	GET `/v1/entries?status=unread` with an `X-Auth-Token` header
3	**Split Entries**	`splitOut`	Splits the `entries` array into one item per article
4	**Screen**	`mistralAi` (Small)	Cheap relevance assessment, replies with structured JSON
5	**Relevant?**	`if`	Parses the screening JSON and branches on `relevant`
6	**Enrich**	`mistralAi` (Large)	Summarises relevant articles into an intel summary + IOCs
7	**Mark Read**	`httpRequest`	PUT `/v1/entries` sets status `read`

Tune the screening prompt: Node 4's system prompt decides the quality. Be specific about what "relevant" means to you (e.g. only actively exploited vulnerabilities, only certain sectors) — the sharper the prompt, the less noise slips through to the expensive Enrich node.

→Importable JSON

n8n import

Copy the below and paste into n8n via **Workflows → ⋯ → Import from clipboard**. Replace the credential references and prompts with your own.

news_workflow.json

```
{
  "name": "Miniflux News Screening",
  "nodes": [
    {
      "parameters": {
        "rule": {
          "interval": [
            {
              "field": "hours",
              "hoursInterval": 1
            }
```

```
          ]
        }
      },
      "name": "Schedule Trigger",
      "type": "n8n-nodes-base.scheduleTrigger",
      "typeVersion": 1.1,
      "position": [
        240,
        300
      ]
    },
    {
      "parameters": {
        "url": "=https://rss.defencia.dk/v1/entries?status=unread&limit=100&direction=asc",
        "options": {
          "response": {
            "response": {
              "fullResponse": false
            }
          }
        },
        "sendHeaders": true,
        "headerParameters": {
          "parameters": [
            {
              "name": "X-Auth-Token",
              "value": "={{ $credentials.minifluxToken }}"
            }
          ]
        }
      },
      "name": "Get Unread (Miniflux)",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 4.2,
      "position": [
        460,
        300
      ]
    },
    {
      "parameters": {
        "fieldToSplitOut": "entries",
        "options": {}
      },
      "name": "Split Entries",
      "type": "n8n-nodes-base.splitOut",
      "typeVersion": 1,
      "position": [
        680,
        300
      ]
    },
    {
      "parameters": {
        "modelId": "mistral-small-latest",
        "messages": {
          "values": [
            {
              "role": "system",
              "content": "You are a security analyst. Assess whether the article is relevant for threat monitoring. Reply ONLY with JSON:
{\"relevant\": true/false, \"category\": \"...\", \"reason\": \"...\"}"
            },
            {
              "role": "user",
              "content": "=Title: {{ $json.title }}\nContent: {{ $json.content }}"
            }
          ]
        }
      },
      "name": "Screen (Mistral Small)",
      "type": "n8n-nodes-base.mistralAi",
      "typeVersion": 1,
      "position": [
        900,
        300
      ]
    },
    {
      "parameters": {
        "conditions": {
          "options": {
            "caseSensitive": true
          },
```

```
        "conditions": [
          {
            "leftValue": "={{ JSON.parse($json.choices[0].message.content).relevant }}",
            "rightValue": true,
            "operator": {
              "type": "boolean",
              "operation": "true"
            }
          }
        ]
      }
    },
    "name": "Relevant?",
    "type": "n8n-nodes-base.if",
    "typeVersion": 2,
    "position": [
      1120,
      300
    ]
  },
  {
    "parameters": {
      "modelId": "mistral-large-latest",
      "messages": {
        "values": [
          {
            "role": "system",
            "content": "Summarise the article into a short threat-intel summary with IOCs if present."
          },
          {
            "role": "user",
            "content": "={{ $json.title }}\n{{ $json.content }}"
          }
        ]
      }
    },
    "name": "Enrich (Mistral Large)",
    "type": "n8n-nodes-base.mistralAi",
    "typeVersion": 1,
    "position": [
      1340,
      220
    ]
  },
  {
    "parameters": {
      "method": "PUT",
      "url": "=https://rss.defencia.dk/v1/entries",
      "sendHeaders": true,
      "headerParameters": {
        "parameters": [
          {
            "name": "X-Auth-Token",
            "value": "={{ $credentials.minifluxToken }}"
          }
        ]
      },
      "sendBody": true,
      "jsonBody": "={ \"entry_ids\": [ {{ $json.id }} ], \"status\": \"read\" }"
    },
    "name": "Mark Read (Miniflux)",
    "type": "n8n-nodes-base.httpRequest",
    "typeVersion": 4.2,
    "position": [
      1560,
      300
    ]
  }
],
"connections": {
  "Schedule Trigger": {
    "main": [
      [
        {
          "node": "Get Unread (Miniflux)",
          "type": "main",
          "index": 0
        }
      ]
    ]
  },
  "Get Unread (Miniflux)": {
    "main": [
```

```
        [
          {
            "node": "Split Entries",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Split Entries": {
      "main": [
        [
          {
            "node": "Screen (Mistral Small)",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Screen (Mistral Small)": {
      "main": [
        [
          {
            "node": "Relevant?",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Relevant?": {
      "main": [
        [
          {
            "node": "Enrich (Mistral Large)",
            "type": "main",
            "index": 0
          }
        ],
        [
          {
            "node": "Mark Read (Miniflux)",
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Enrich (Mistral Large)": {
      "main": [
        [
          {
            "node": "Mark Read (Miniflux)",
            "type": "main",
            "index": 0
          }
        ]
      ]
    }
  },
  "settings": {
    "executionOrder": "v1"
  }
}
```

After import — always check: the credential bindings (Mistral + an optional generic header-auth for the Miniflux token), that node type versions match your n8n, and that the API URLs point to `rss.defencia.dk`. The embedded token reference is a placeholder — bind it to a real n8n credential, never leave it in cleartext.

→Credentials

Credential	Setup
Miniflux token	Generate in Miniflux (Settings → API Keys). Store as an n8n header-auth credential or env variable — not inline in the node
Mistral API	n8n's Mistral credential with your API key; used by both the Screen and Enrich nodes

→Tuning & extensions

Adjustment	Effect
Interval	Lower = fresher news, but more API calls. 1h is a good balance for many feeds
limit	Raise if feeds pile up between runs; watch out for Mistral rate limits
Batching	Screen several articles in one Mistral call to save tokens (requires an array prompt)
Output	Add a node after Enrich: send to a dashboard, Slack, email or a database table
IOC extraction	Add a regex/Code node that pulls IPs, domains and hashes out of the summary
Dedup	Miniflux's read status is your dedup — which is why everything is marked read in both branches

Link to the CVE flow: This handles news/articles. Vulnerabilities follow a separate pattern via OpenCVE webhooks — see the CVE enrichment workflow.