Building Security Monitoring Solutions with Open Souce Tools

The Nineteenth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2025)

October 26, 2025 to October 30, 2025 - Barcelona, Spain

Ali Recai Yekta (ali@yekta-it.de) , Yekta IT GmbH

whoami

Ali Recai Yekta

CTO & Head of Cybersecurity

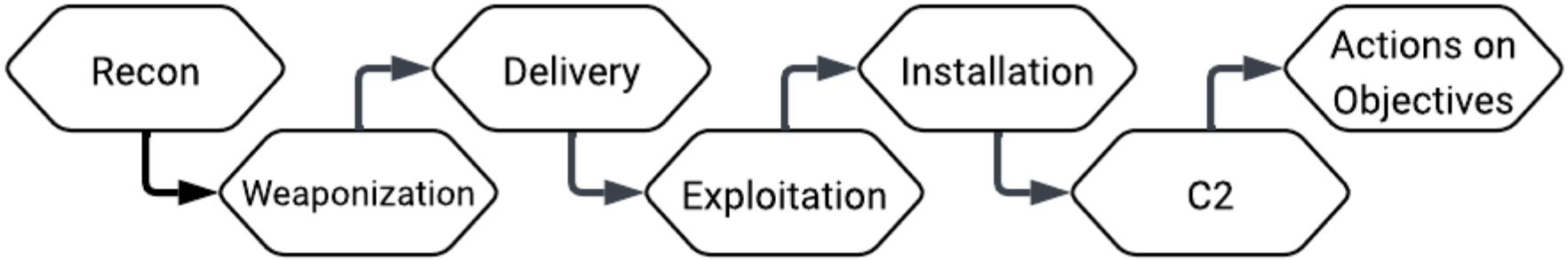
- 15+ years of cybersecurity experience in IT and OT systems
- Red and Blue Team specialist
- Building and operating SOCs for critical infrastructure
- Extensive experience in railway, automotive, and energy sectors
- Co-founder and Head of Cybersecurity at Yekta IT GmbH
- Research focus on OT security
- Master's degree in IT Security (Ruhr University Bochum)
- OSCP, OSWE, OSEP, CRTO Certifications

Topics

- Attack Methdology
- Security Monitoring
- Collection
- Analysis
- Respond
- Demo

Attack Methodology

Cyber Kill Chain



- **Recon:** Research, identifications, target selection
- **Weaponization:** Malware or exploits are created for the target
- **Delivery:** Malicious payload is sent via phishing, web, USB, etc.
- **Exploitation:** The weapon's code is triggered, exploiting flaws in the system
- **Installation:** The weapon installs a backdoor to allow persistent access
- **Command and Control:** Command channel for remote manipulation of the target
- **Actions on Objectives:** Attackers achieve their goal, e.g. data theft or disruption

MITRE ATT&CK

https://attack.mitre.org

What it is:

- It's a public **knowledge base** of real-world adversary tactics and techniques.

How it's organized:

- **Tactics** = the "why"
- **Techniques** = the "how" (with sub-techniques)
- **Defensive content included**: each technique page links Mitigations (preventive controls) and Detection guidance (what to log/look for)
- Scope: **Enterprise** (Windows/macOS/Linux, cloud, network devices, containers, …), **Mobile** and **ICS**

MITRE ATT&CK

Why

How

VATT&EK

What it is:

- **Vehicle Adversarial Tactics, Techniques & Expert Knowledge** a specific taxonomy to formalize attacks on Intelligent Transport Systems (ITS).

Why now:

- Automotive & rail are highly networked, heterogeneous, and hard to secure; generic frameworks (e.g., ATT&CK) miss ITS specifics. **VATT&EK fills the gap**.
- Automotive case **Jeep Hack 2015**:
  Exploitation of Internet-Accessible Device → Inter-process Communication → Reprogram ECU (Privilege Escalation) → (Control Lights | Kill Engine | Control IPC | Control Brake)
- Rail case **Poland trains stop**:
  Tactic: **Affect Vehicle Function** with a **simple radio hack**
  Procedure: sending radio signals to trigger **emergency stop**

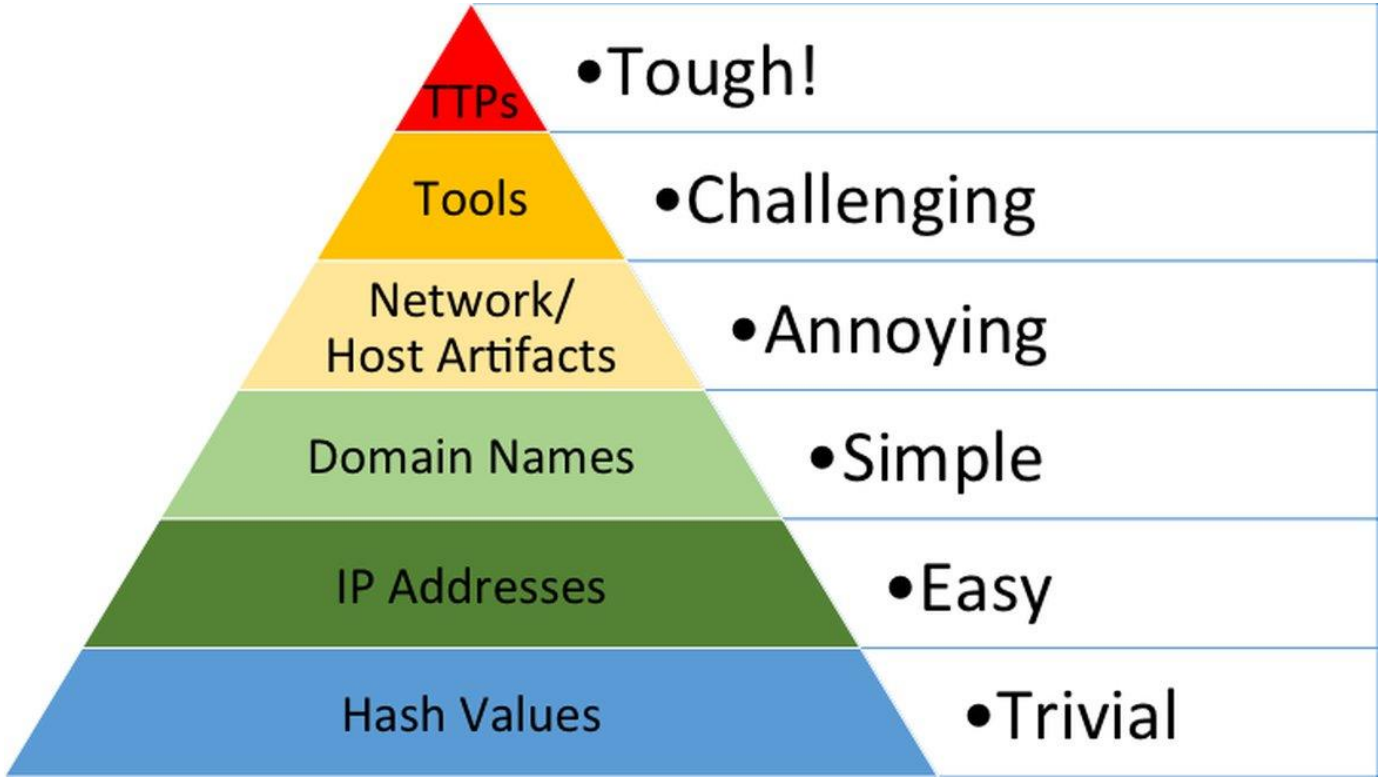
VATT&EK

- Building blocks:
  Tactics (attacker goals)
  Techniques (how)
  Procedures (step-by-step)
  Expert Knowledge (OEM/ops insights)
- Scope today: **14 tactics** and **127 techniques, 78 of them new** to the domain.

A few use case examples:

- **TARA (threat analysis & risk):** map assets → VATT&EK techniques → prioritize mitigations.
- **Pentesting:** design scenario-based tests per technique
- **Threat intel:** tag reports to techniques
- **IR/Training:** reconstruct paths; shared language across teams

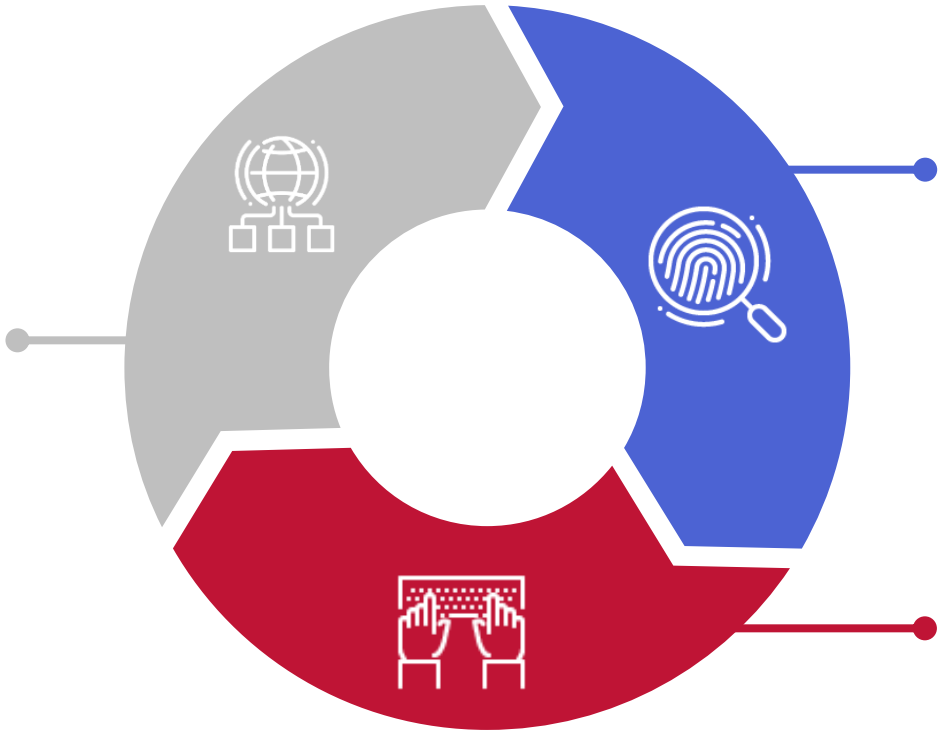
Pyramid of Pain



The Security Monitoring Cycle

Collection

Collecting log data and alerts from various sources



Analytics

Responding to incidents and initiating countermeasures

Respond

Responding to incidents and initiating countermeasures

Collection

Sensors

What are security sensors?

- Tools that **observe systems or networks** and report suspicious activity
- Common goals: **see traffic, find threats, give context**
- They don’t fix issues by themselves! They **collect, detect, and inform**
- Examples: **Zeek** (network analytics), **Suricata** (IDS/IPS), **YARA** (file/memory scanning), …

Sensors

Where they sit & what they see:

- **Network sensors:** packets, protocols, files in transit
- **Host sensors**: processes, files, memory, logs
- **Log sensors**: auth logs, DNS, proxy, cloud logs

Zeek

What Zeek is (network security monitor):

- **Event-based** network analyser; **passive** (not inline)
- **Automatic protocol detection** + **protocol parsers**
- Works on **live traffic** or **PCAP**
- **Multi-threading** for high throughput

Zeek

What Zeek produces:

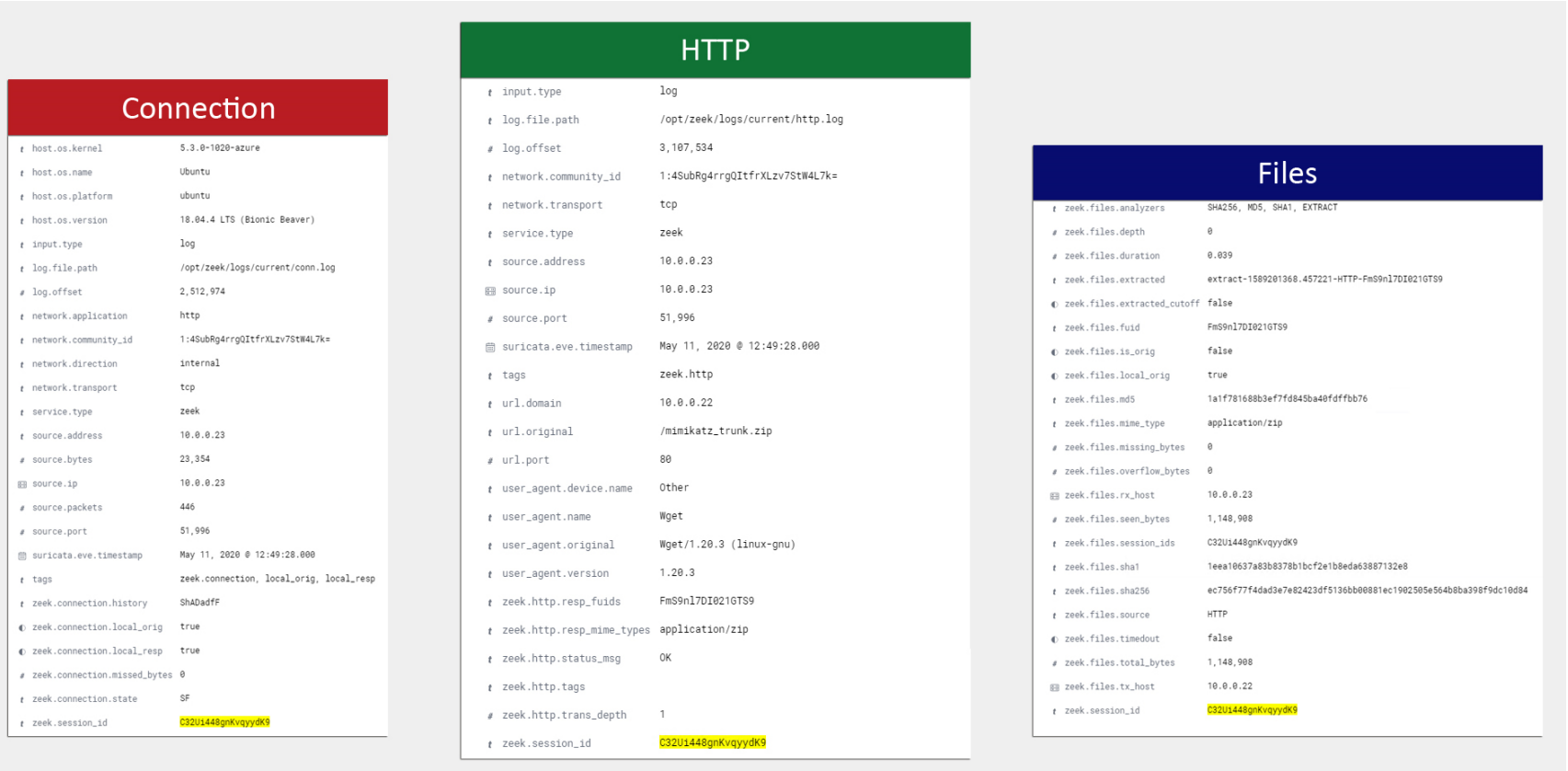
- Rich **traffic logging** (HTTP, DNS, SSL/TLS, files, etc.)
- **JSON** (and TSV) outputs that are easy to ingest
- **File extraction** from flows for later analysis
- Built-in **Threat Intelligence** framework (match lists/feeds)
- **Custom scripting** with Zeek's language (formerly **Bro**) to add logic

Zeek

Detection with Zeek:

- Great for **behavioural detection** and **context** (“who talked to whom, how, when”)
- **Intrusion Detection** via scripts & intel matches
- **Prevention** requires **integration** (e.g., SIEM → firewall block) Zeek itself is not an inline IPS
- Best for: broad visibility, investigations, and feeding other tools with high quality metadata

Zeek



Connection

host.os.kernel	5.3.0-1020-azure
host.os.name	Ubuntu
host.os.platform	ubuntu
host.os.version	18.04.4 LTS (Bionic Beaver)
input.type	log
log.file.path	/opt/zeek/logs/current/conn.log
log.offset	2,512,974
network.application	http
network.community_id	1:4SubRg4rrgQItfrXLzv7StW4L7k=
network.direction	internal
network.transport	tcp
service.type	zeek
source.address	10.0.0.23
source.bytes	23,354
source.ip	10.0.0.23
source.packets	446
source.port	51,996
suricata.eve.timestamp	May 11, 2020 @ 12:49:28.000
tags	zeek.connection, local_orig, local_resp
zeek.connection.history	ShADadfF
zeek.connection.local_orig	true
zeek.connection.local_resp	true
zeek.connection.missed_bytes	0
zeek.connection.state	SF
zeek.session_id	C32Ui448gnKvqyydK9

HTTP

input.type	log
log.file.path	/opt/zeek/logs/current/http.log
log.offset	3,107,534
network.community_id	1:4SubRg4rrgQItfrXLzv7StW4L7k=
network.transport	tcp
service.type	zeek
source.address	10.0.0.23
source.ip	10.0.0.23
source.port	51,996
suricata.eve.timestamp	May 11, 2020 @ 12:49:28.000
tags	zeek.http
url.domain	10.0.0.22
url.original	/mimikatz_trunk.zip
url.port	80
user_agent.device.name	Other
user_agent.name	Wget
user_agent.original	Wget/1.20.3 (linux-gnu)
user_agent.version	1.20.3
zeek.http.resp_fuids	FmS9nl7DI021GTS9
zeek.http.resp_mime_types	application/zip
zeek.http.status_msg	OK
zeek.http.tags	
zeek.http.trans_depth	1
zeek.session_id	C32Ui448gnKvqyydK9

Files

zeek.files.analyzers	SHA256, MD5, SHA1, EXTRACT
zeek.files.depth	0
zeek.files.duration	0.039
zeek.files.extracted	extract-1589201368.457221-HTTP-FmS9nl7DI021GTS9
zeek.files.extracted_cutoff	false
zeek.files.fuid	FmS9nl7DI021GTS9
zeek.files.is_orig	false
zeek.files.local_orig	true
zeek.files.md5	1a1f781688b3ef7fd845ba40fdffbb76
zeek.files.mime_type	application/zip
zeek.files.missing_bytes	0
zeek.files.overflow_bytes	0
zeek.files.rx_host	10.0.0.23
zeek.files.seen_bytes	1,148,908
zeek.files.session_ids	C32Ui448gnKvqyydK9
zeek.files.sha1	1eea10637a83b8378b1bcf2e1b8eda63887132e8
zeek.files.sha256	ec756f77f4dad3e7e82423df5136bb00881ec1902505e564b8ba398f9dc10d84
zeek.files.source	HTTP
zeek.files.timedout	false
zeek.files.total_bytes	1,148,908
zeek.files.tx_host	10.0.0.22
zeek.session_id	C32Ui448gnKvqyydK9

Suricata

What Suricata is (IDS/IPS/NSM):

- **Intrusion Detection / Prevention** (IDS/IPS), plus network security monitoring
- Runs on **live links** or **PCAP**; can be **inline** to block
- **Multi-threading**; designed for speed on modern CPUs

Suricata

Protocols, parsing & outputs:

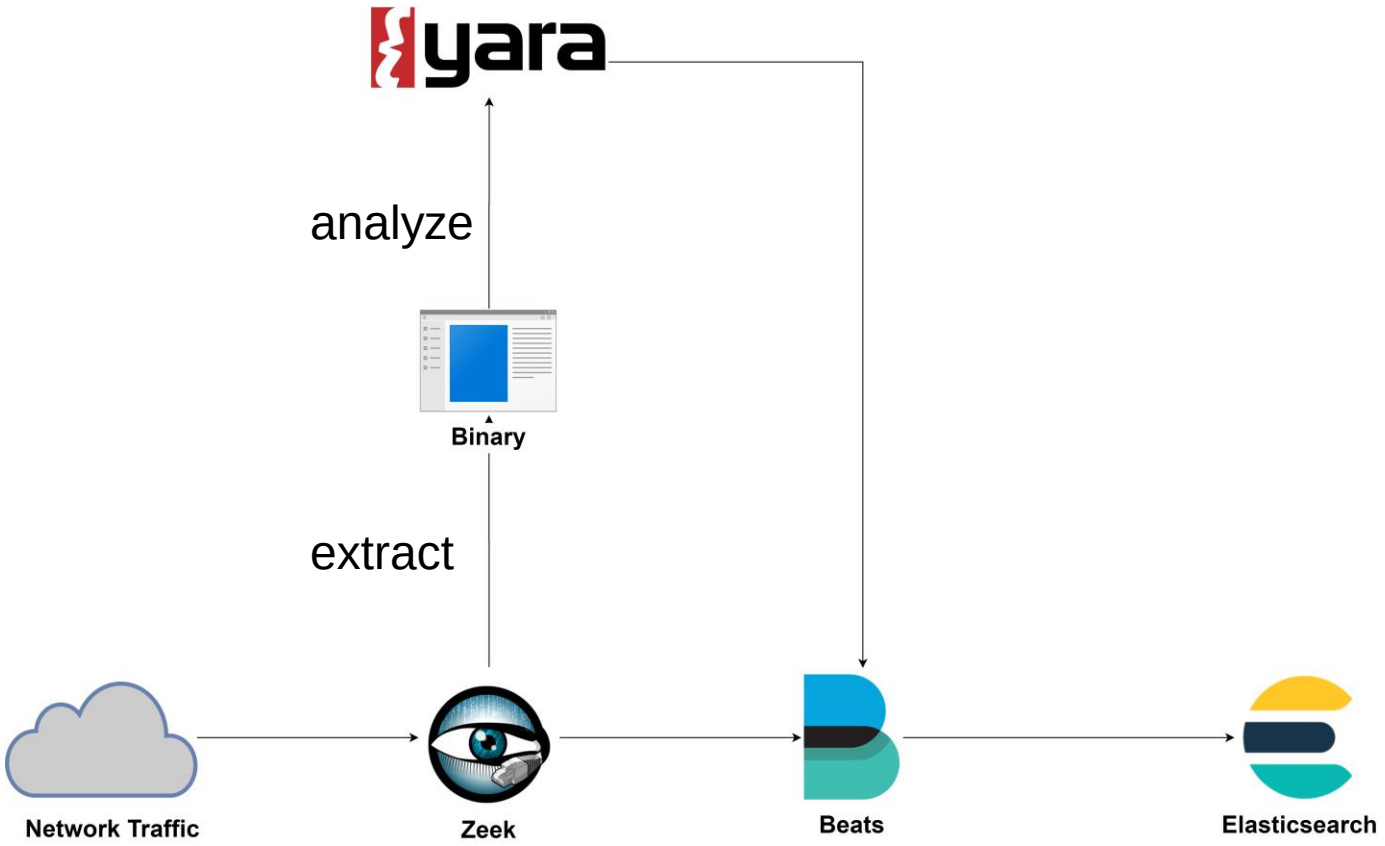
- **Automatic protocol detection** + **protocol parsers**
- **Rules-driven** detection
- **Traffic logging** and **JSON output**
- **File extraction** for downstream scanning
- **IP reputation** lists and **Lua** scripting for advanced logic

Suricata

Rules & practical use:

- Use public/commercial rule sets **(keep them updated)** + **custom rules**
- Tune to your network to reduce **false positives**
- In **IDS mode**: alert only; in **IPS mode**: block specific threats
- Great for **known-bad** detection and **real-time blocking**

YARA


yara
analyze
Binary
extract
Network Traffic
Zeek
Beats
Elasticsearch

YARA

What YARA is:

- A **rule-based pattern matcher** for **files and memory**
- Finds malware or artifacts using **strings, regex, and conditions**
- Works on disk, memory dumps, or extracted files

YARA

How to use it:

- Write simple **rules** (name, meta, strings, condition)
- Scan **suspicious files, email attachments, extracted payloads**

Good rules = good results:

- Add **metadata** (author, source, reference)
- Prefer **stable strings** (avoid unreliable indicators)
- Keep rules **efficient** (limit heavy regex, test before production)
- Organize rules by **category**; version control them

Windows (Events)

What Windows Event Logs are:

- Windows keeps **structured logs** about actions and system activity.
- Each record is called an **Event**, identified by an **Event ID** (a numeric code).
- Event logs are grouped by **channels**:
  - **Security:** logons, privilege use, policy changes
  - **System:** driver, service, or kernel-level events
  - **Application:** messages from user apps or services
  - **Other channels** (optional): PowerShell, Sysmon, DNS Client, etc.
- Each event contains:
  - **Event ID**, **timestamp**, **user**, **process**, and **computer name**

Windows (Events)

How to use Windows Events in Security Monitoring:

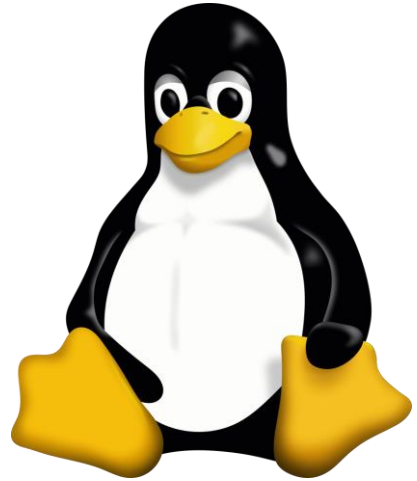
- Collect logs via **Windows Event Forwarding (WEF)** or an **agent** (e.g., Winlogbeat, Wazuh, Splunk UF).
- Forward to your **SIEM** or central log system.
- Create **filters or alerts** on specific Event IDs or combinations:
  - ID 4625 (failed logins) + same IP → brute-force
  - ID 4688 (process create) + unusual path → suspicious execution
- Combine with **Zeek/Suricata** network data for even better detection coverage.
- Store logs as **EVTX** (native) or **JSON** (exported) for analysis.

Sysmon

What Sysmon Is:

- **Sysmon** = *System Monitor*, part of Microsoft's **Sysinternals Suite**
- Runs as a **Windows service and driver**
- Extends Windows event logging with **detailed system activity**
- Events can be found in **Windows Event Log → Applications and Services Logs → Microsoft → Windows → Sysmon Operational**
- Focuses on: **Process, file, registry, and network activity** that normal Event Logs often miss

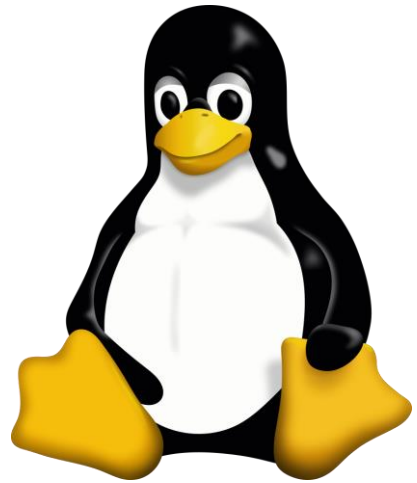
Linux (Auditd)



What auditd is

- **auditd** = Linux **Audit Daemon**
- Monitors **system calls** and **security-relevant actions**
- Built into most distros (**RHEL, Ubuntu, Debian, etc.**)
- Logs go to /var/log/audit/audit.log

Linux (Auditd)



What auditd records

- **User actions** (logins, sudo, file access, command execution)
- **System changes** (config files, binaries, permissions)
- **Process creation** (fork, execve, etc.)
- **File monitoring** via -w rules (watch specific files or directories)

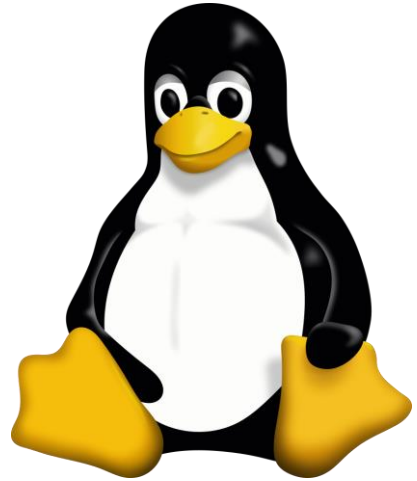
auditctl -w /etc/passwd -p wa -k passwd_changes

-p = permissions
-k = key

→ watches for writes/attribute changes to /etc/passwd

ausearch -k passwd_changes

Linux (Auditd)



auditd in practice:

- Use with **audispd** or **Filebeat** to forward logs
- Tune rules; too many → too much noise
- Good starting events:
  - USER_LOGIN, EXECVE, CHMOD, CHOWN, ADD_USER, DEL_USER
- Combine with network sensors → see both **system activity + network behavior**
- Ideal for **host-level visibility** and **incident investigation**

Img. Source:
https://www.kernel.org/

Analytics

Elasticsearch

What it is:

- **Elasticsearch**: a **distributed JSON datastore + search/analytics engine**
  - You store **events as JSON documents**
  - Data is grouped in **indexes**, split into **shards**, with **replicas** for speed & resilience
  - Great at **filtering** (exact matches, time ranges) and **aggregations** (counts, top N, histograms)
- **Kibana**: the **web UI** for Elasticsearch
  - Search logs, pivot across fields, build **dashboards** and **alerts**
- Why security teams use it: handles **large volumes of time-stamped events**, lets you **search and summarize quickly**

Img. Source: https://www.elastic.co/products/elasticsearch

Elasticsearch

How can we use it:

- Treat each log line as one **JSON document**
- **Ask questions as filters/aggregations**
  - "Show **failed logons** in the last 24h by **username**"
  - "Top **destination IPs** for this host today"
  - "Find **rare user-agents**"
- Build **dashboards** (trend lines, maps) and **alerts** (thresholds, anomaly spikes)
- Manage data over time (keep hot recent data, move old data to cheaper storage or delete)

Img. Source:
https://www.elastic.co/products/elasticsearch

Elasticsearch

Example security pipeline:

Flow: Zeek / Suricata → (Beats or Elastic Agent) → [optional Logstash] → **Elasticsearch** → **Kibana**

Img. Source:
https://www.elastic.co/products/elasticsearch

Threat Intelligence

What it is:

- **Threat Intelligence = actionable knowledge** about attackers that helps you **detect, block, and investigate**
  - Two parts:
    1. **IOCs (Indicators of Compromise):** concrete clues you can match in data
    2. **TTPs (Tactics, Techniques, Procedures): behaviours** attackers use (e.g. ATT&CK)

Why it matters:

- find threats earlier, cut false positives with context, speed up response

Sources: your own detections, sharing communities, vendor feeds, public reports

Threat Intelligence

- **Indicator types:**
  Indicator = a data clue that points to attacker activity. Types are grouped by *what the clue points to*.
- **Infrastructure indicators → attacker servers/services**
  - Examples: **IP**, **domain**, **URL**, **certificate fingerprint**, **TLS client/server fingerprint**
- **File/artifact indicators → specific malware or tools**
  - Examples: **hashes** (MD5/SHA256), **YARA rule matches**, known **file names/paths**
- **Host-behavior indicators → suspicious actions on a machine**
  - Examples: **process + command line**, **registry keys**, **scheduled tasks/services**, **persistence locations**
- **Network-behavior indicators → how it behaves on the wire**
  - Examples: rare **user-agents**, **DNS patterns**, unusual **protocol use**
- **Phishing/campaign indicators → social engineering clues**
  - Examples: **sender address**, **subject lines**, **attachment hashes**, **malicious links**
- *Rule of thumb:* single, fragile clues (IP/hash) **expire fast** combine multiple indicators or use **behavior (TTPs)** for resilience.

Threat Intelligence

How Threat Intel can be used in our sample stack:

- **Suricata:** load **signature rules** + **IP/domain reputation** → alert or **block** known-bad
- **Zeek:** When Zeek logs known-bad **IPs, domains, and file hashes** in traffic, it **marks the related log entry** so you can **alert or search** on it.
- **YARA:** scan **extracted files** or repositories using intel-derived rules
- **Elasticsearch: enrich** events with Intel (add fields like intel.match, confidence, first_seen)
- **Kibana: alert** on matches, **pivot** to related Zeek flows/Sysmon events

Respond

TheHive

What it is:

- The Hive is an **Open-source Incident Response platform**
- It turns **alerts** into structured **cases** with tasks, evidence & timeline
- Coordinates teams
- Integrates with **Cortex** for **automation**
- Sits after detection (Elastic/IDS/Zeek) and before action (Cortex)

TheHive

Core concepts:

- **Alert**: signal from SIEM, IDS or EDR that needs triage, can auto-create a case
- **Observables**: things to check such as hash, IP, URL or domain, sent to Cortex analyzers automatically, results are added to the case
- **Tasks**: steps with owner and due time, responders can be launched from here to take action
- **Timeline**: automatic clear log of notes, actions, results and attachments for hand-offs and audit

Cortex

What it is:

- Analysis & action engine connected to TheHive
- **Analyzers:** enrich observables (e.g. YARA)
- **Responders:** take actions (block IOCs, quarantine files, delete emails)
- **Extras:** API-first, custom Python analyzers/responders, key mgmt, rate-limiting, result caching

TheHive + Cortex

Automated Malware Analysis & Removal:

- **Flow**: Alert → TheHive case → Observables → Cortex analyzers
- **Decisioning**: scoring/tags trigger tasks or auto-actions
- **Remediation via responders**:
  - Kill process / quarantine or delete malicious file
  - Block hash/domain/IP
  - Open/close tickets, notify stakeholders

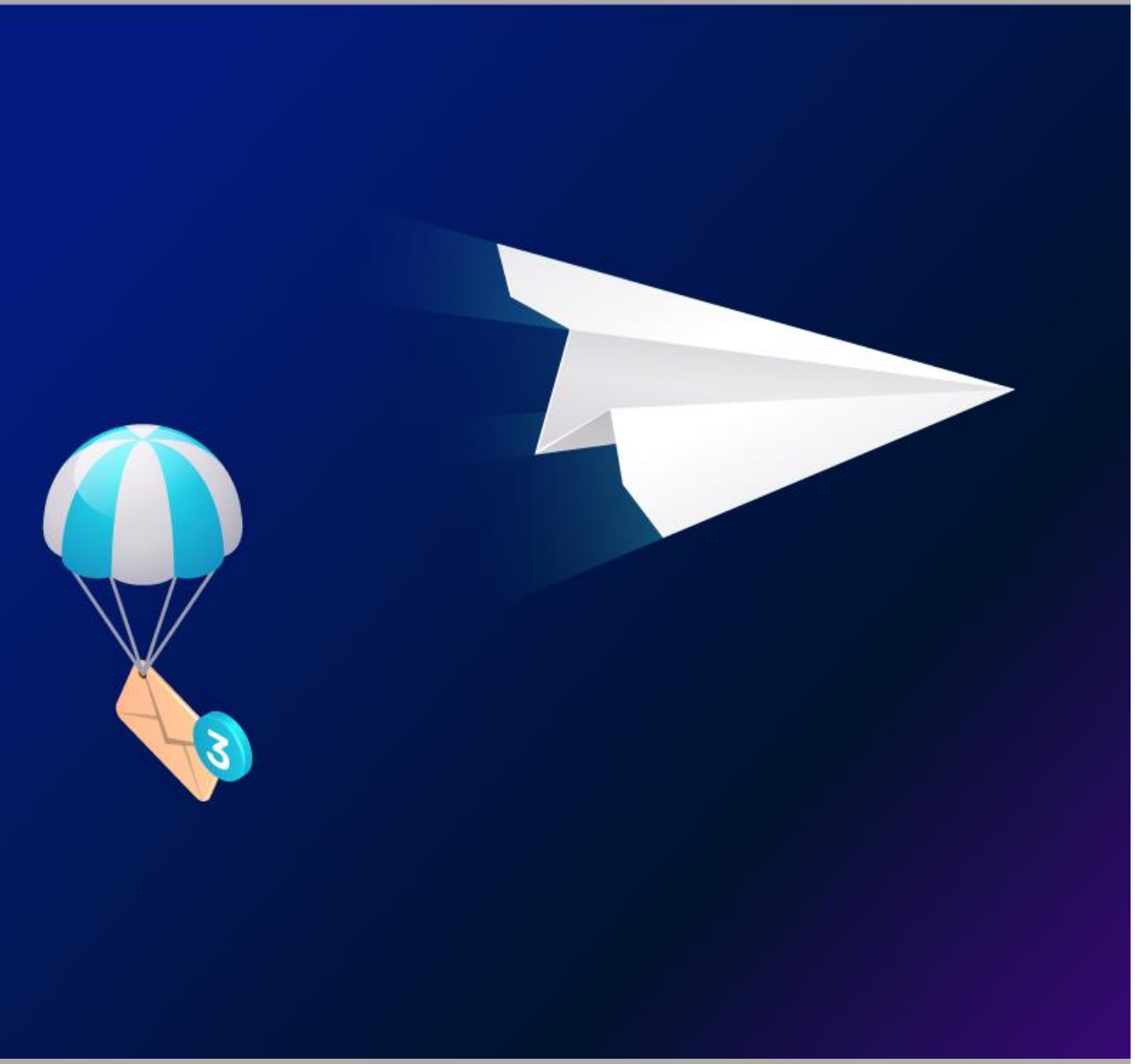
TheHive + Cortex

Host Isolation:

- **Flow:** Alert → **TheHive** case → Observables (host/IP/EDR id) → **Cortex** analyzers
- **Decisioning:** tags / score / playbook match → trigger **Isolate host** task or a similar action
- **Action (responders):** Network Access Control quarantine, firewall blocklist, … → results added to case timeline
- **Notify & evidence:** owner + SOC notified, analyzer verifies isolation, all actions are logged

Demo

Thank you



Contact

Ali Recai Yekta

ali@yekta-it.de

Yekta IT GmbH

www.yekta-it.de