



UNIVERSITÉ LIBRE DE BRUXELLES

Honeypots, honeynets & AI-Based anomaly detection

Implementation Project — Network Security

Group members

DELCROIX Brice

GODRIE Clément

PISCAGLIA Timothé

CORMONTAGNE Romain

MEEUS Dimitri

Professor Jean-Michel Dricot

Academic Year 2025–2026

Contents

1	Introduction and scope	2
2	Theoretical Background	2
2.1	What is a honeypot ?	2
2.2	Honeynets	3
2.3	Deception-Based Defence	3
3	Architecture	3
3.1	T-Pot platform overview	3
3.2	Network Architecture	4
4	Deployment & return of experience	5
4.1	Infrastructure setup	5
4.2	Installation process	5
4.3	Resource Consumption : Observed vs. Theoretical	5
4.4	Key configuration decisions	6
5	Data Collected	6
5.1	Scale of attacks	6
5.2	Geographic distribution	8
5.3	Most targeted ports	8
6	Machine Learning — Anomaly Detection	9
6.1	Problem framing	9
6.2	Algorithm choice : Isolation Forest	9
6.3	Feature engineering	9
6.4	Model configuration	10
6.5	Results	10
6.6	Heatmap analysis : Type x IP reputation	11
6.7	Top suspicious connections	11
7	Relevance to today's Network Security	12
7.1	Threat intelligence	12
7.2	AI integration in modern SOC's	12
7.3	Limitations	12
8	AI strategy	13
8.1	Overview	13
8.2	Use cases	13
8.3	Correctness assessment	13
8.4	Prompting optimisation	13
8.5	Individual reflections	14
9	Conclusion	14

1 Introduction and scope

This report presents our network security examination project conducted as part of the ULB Network Security Course. We chose to do an implementation. Our work consists of a complete, working deployment of a honeynet infrastructure exposed to real internet traffic, combined with a machine learning model designed to detect anomalies in the collected attack data.

The project scope covers four main points:

- Architecture explanation : the T-Pot honeynet platform, its components, and the ELK stack
- Return of experience : concrete deployment challenges, configuration decisions, and lessons learned
- A live demo : real-time Kibana dashboard showing attack traffic and ML anomaly detection results
- Relevance to today's network security : why honeypots matter in modern security architectures

The honeynet was deployed on a dedicated VPS and remained active for approximately two weeks before the data collection. During this period, the system collected over 30 million attack events originating from across the globe. A representative sample of 54,124 entries was used to train an isolation forest anomaly detection model, which successfully identified 2,617 anomalous connections worthy of further investigation.

2 Theoretical Background

2.1 What is a honeypot ?

A honeypot is a security resource whose value lies in being probed, attacked or compromised. Unlike traditional security systems that aim to block intrusions, honeypots are intentionally exposed to attract attackers. Their purpose is to observe, log, and analyze malicious behavior in a controlled environment without putting production systems at risk. Honeypots can be classified along two main axes:

Dimension	Low Interaction	High Interaction
Simulation depth	Emulates services superficially	Full OS and service stack
Data richness	Limited attack data	Rich behavioral data
Risk	Very low	Higher (real OS exposed)
Maintenance	Simple	Complex
Example	Cowrie (SSH emulation)	Full Linux honeypot

Table 1: Comparison between low-interaction and high-interaction honeypots.

2.2 Honeynets

A honeynet is a network of honeypots working together, typically combined with logging infrastructure, network capture tools and visualization dashboard. The goal is to provide a broader view of attack patterns by simulating multiple services simultaneously : SSH servers, web servers, industrial SCADA systems, Windows file shares, and more. T-Pot developed and maintained by *Deutsche Telekom Security*, is the leading open source honeynet platform. It packages over 20 individual honeypots inside docker containers, managed by a single installer and monitored through a unified ELK (Elasticsearch, Logstash, Kibana) stack.

2.3 Deception-Based Defence

Honeypots are part of a broader defensive strategy known as cyber deception. Rather than purely hardening defences, deception based approaches deliberately create false targets to mislead attackers, waste their time and resources, and gather threat intelligence. The MITRE ENGAGE framework formalizes these techniques, classifying honeypots as a core deception tool alongside fake credentials, decoy documents and simulated network services. In modern security operations centers, honeynet data feeds into threat intelligence pipelines used for enriching indicators of compromise (IOC) databases, identifying scanning infrastructure, and providing early warning of emerging attack patterns before they hit production systems.

3 Architecture

3.1 T-Pot platform overview

T-Pot orchestrates all honeypot services through Docker Compose. Each honeypot runs in an isolated container, sharing only the host network interface. This architecture provides several advantages : honeypots cannot interfere with each other, containers can be updated independently, and restarting them resets the entire platform to a clean state. The below table provides an overview of the different decoy services that T-Pot has to offer.

Component	Role	Port(s)
Medpot	Simulates medical HL7/MLLP protocol services	2575
Honeypots (qHoney-pots)	Simulates 30+ services: FTP, HTTP, RDP, SMB, SMTP...	21, 25, 80, 443, 445, 3389...
Honeytrap	Dynamic TCP honeypot : catches attacks on any port	Dynamic
CiscoASA	Simulates Cisco firewall login portal	5000/UDP, 8443
ConPot	Simulates SCADA/ICS industrial systems	102, 161, 502, 623, 2404
Dicompot	Simulates medical DICOM imaging systems	104, 11112
ADBhoney	Simulates Android Debug Bridge	5555
Suricata	Network IDS : detects attack signatures	All traffic
P0f	Passive OS fingerprinting	All traffic
FATT	Fingerprint all the things : TLS/SSH fingerprinting	All traffic
Elasticsearch	Log storage and indexing	64298 (internal)
Logstash	Log collection and parsing pipeline	64305 (internal)
Kibana	Web dashboard for visualization	64297
NGINX	Reverse proxy for web interface	64294, 64297

Table 2: T-Pot Overview

3.2 Network Architecture

The VPS exposes all honeypot ports directly to the internet. The real SSH management port is moved to 64295 by the T-Pot installer to avoid conflict with the honeypots running on port 22. The Kibana dashboard is accessible on port 64297 and protected by HTTP Basic Authentication.

Layer	Details
VPS Host	Ubuntu 24.04 LTS — OUI Heberg Marseille — 12GB RAM, 70GB NVMe, 4 vCores
Container Runtime	Docker + Docker Compose (managed by T-Pot)
Honeypot Ports	Ports 21–23, 25, 53, 80, 443, 445, 3389... exposed to 0.0.0.0
Management SSH	Port 64295 — restricted to team members
Web Dashboard	Port 64297 — NGINX reverse proxy → Kibana

Table 3: Network architecture layers and exposed services.

4 Deployment & return of experience

4.1 Infrastructure setup

We first tested the deployment on local VirtualBox VM running Ubuntu 24.04 Server with 8 GB RAM and 80 GB virtual disk before committing to a paid VPS. This test phase revealed the real resource consumption of T-Pot Mini : approximately 13GB disk and 7.7GB RAM after startup — confirming that the minimum 8GB RAM requirement from the official documentation is accurate but leaves very little headroom.

Based on our VM measurements, we selected the OuiHeberg VPS Linux 12G offer (12GB RAM, 70GB NVMe SSD, 4 vCores) as it provided comfortable margins above the minimum requirements. The VPS was provisioned with Ubuntu 24.04 and delivered within 60 seconds.

4.2 Installation process

The T-Pot installation uses an Ansible playbook triggered by a single curl command. The process installs Docker, pulls all container images (20 images, several GB), and configures the system. Total installation time was approximately 90 minutes, dominated by image downloads.

We encountered some significant issues during installation and utilisation:

- **Kibana crashes due to RAM saturation** : Elasticsearch and Kibana together consumed 8.2GB on an 11GB VPS, causing Kibana to restart repeatedly. Mitigated by adding a 4GB swapfile.
- **Elasticsearch 10,000 result export limit** : bulk data export failed because ES limits queries to 10,000 results. Required per-index day-by-day queries to export the full dataset.
- **VPS's own IP polluting anomaly results** : Suricata logs outgoing connections from the server itself, causing the VPS IP to appear as the top "suspicious" source. Required filtering before ML training.
- **Kibana CSV export failure** : the built-in Kibana export returned a `ReportingError: Failed to decrypt report job data error`. Had to fall back to direct Elasticsearch API queries.

4.3 Resource Consumption : Observed vs. Theoretical

Metric	T-Pot Official Spec (Mini)	Observed on VM	Observed on VPS
RAM	8GB minimum	7.7GB used / 7.1 total	8.2 GB used / 11 GB total
Disk (after install)	128GB recommended for sensor (no official number for Mini)	14 GB used / 79 GB total	13 GB used / 67 GB total
Swap	Not specified	None (risky)	4GB added manually
CPU	Not specified	20% average	15% average

Table 4: Resource consumption.

The RAM usage was higher than expected after a few days of operation, causing kibana to crash and restart. We mitigated this by adding a 4 GB swapfile, which provides a safety margin for Elasticsearch’s memory-intensive indexing operations.

4.4 Key configuration decisions

- T-Pot type : We chose Mini mode (not standard/Hive) to reduce resource consumption while retaining all core honeypots and the full ELK stack.
- Swap : Added 4 GB swapfile after observing RAM saturation : made persistent via `/etc/fstab`.
- SSH port : T-Pot automatically moves SSH to port 64295.
- Data export : Elasticsearch’s 10000 result query limit required us to query per day indices separately and combine the results, rather than using a single bulk export.

5 Data Collected

5.1 Scale of attacks

Over approximately 13 days of exposure (May 28 to June 10, 2026), our honeynet collected over 30 million attack events — approximately 2.3 million per day. The first attacks arrived within minutes of the ports being opened, confirming that internet-exposed servers are continuously scanned by automated bots without any manual targeting.

The dataset contains events from several components of the platform:

Honeypot	Events (24h peak)	Primary Attack Type
Honeypots (qHoneypots)	384,000	Multi-service scanning: RDP, SMB, HTTP, FTP
Honeytrap	28,000	Dynamic port scanning — unusual port probes
ConPot	428	SCADA/ICS protocol exploitation attempts
Dicompot	185	Medical system enumeration
CiscoASA	132	VPN/firewall login brute force
ADBhoney	121	Android device exploitation attempts

P0f and Suricata are passive analysis tools — they do not attract attacks but analyse traffic generated by other honeypots.

The exported dataset contains 7 fields: `timestamp`, `type`, `src_ip`, `dest_port`, `src_port`, `protocol` and `ip_rep`. It is not perfectly balanced between all honeypot types. Five components contributed 10,000 events each, while ConPot only contributed 4,124 events — not because of an export limit, but because fewer ConPot events were recorded during the collection period.

For the machine learning analysis, a separate sample was exported from Elasticsearch, with up to 10,000 events per component:

Component	Number of events
Honeypots	10,000
Suricata	10,000
P0f	10,000
Honeytrap	10,000
CiscoASA	10,000
ConPot	4,124

Table 5: Distribution of collected events by T-Pot component.

The sample contains 49,823 events with a source IP address and 3,551 different source IPs. This shows that the honeynet was contacted by a large number of external hosts, not only by a small number of repeated attackers. The use of VPNs from contacting hosts is also to be considered.

The IP reputation field also gives useful information about the nature of the traffic. In the sample, 15k events were associated with known attackers, 2.5k events with mass scanners, and 18 events with bots or crawlers. Most events had no reputation label. This does not mean that the traffic was legitimate, only that no reputation information was available for these IP addresses.

5.2 Geographic distribution

The top attacking countries observed in Kibana over the collection period were the United States, Germany, China, the Netherlands, and Finland. This distribution is typical of internet-wide scanning infrastructure — most traffic originates from cloud providers and hosting companies running automated scanners (Shodan, Censys, Masscan-based tools) rather than from individual human attackers.

The geographic distribution of the attacks shows that the honeynet was exposed to global internet traffic. The source IP addresses came from many different regions, as illustrated by the Kibana attack map.

5.3 Most targeted ports

The most targeted ports also show the type of activity observed. The most frequent destination port was port 67, which mainly corresponds to DHCP. This traffic was not considered as a real external attack source — it originates from the Docker bridge address 172.28.0.1, an internal infrastructure address rather than a real internet attacker. The most important external attack target was port 3306, used by MySQL, with 7,903 events. This suggests automated scanning for exposed database services. Other frequently targeted ports were SNMP on port 161, RDP on port 3389, DNS on port 53, HTTPS on port 443 and SMB on port 445.

Port	Number of events	Interpretation
67	8,786	DHCP-related internal honeypot traffic
3306	7,903	MySQL database scanning
161	2,467	SNMP and monitoring system probing
3389	1,242	RDP scanning
53	912	DNS-related traffic
443	750	HTTPS probing
1025	482	Generic TCP probing on high-numbered port
2404	392	Industrial protocol probing
10001	392	Scanning of less common exposed services
5984	374	CouchDB database probing
623	336	IPMI remote management probing
445	329	SMB / Windows file sharing probing

Table 6: Most targeted destination ports.

One important observation is that the dataset also contained internal or infrastructure-related traffic. For this reason, the preprocessing step had to remove internal IP addresses and rows without valid source IPs before training the model.

6 Machine Learning — Anomaly Detection

6.1 Problem framing

An important clarification about the ML framing: in a honeypot context, all traffic is malicious by definition, there is no legitimate users. The distinction the model makes is therefore not between "attack" and "normal traffic", but between "common automated attack" (banal, repetitive scans seen thousands of times) and "anomalous attack" (unusual behaviour that deviates significantly from the statistical norm and therefore warrants close investigation).

This framing is directly relevant to real-world SOC operations: analysts cannot manually review millions of events. Anomaly detection models are used precisely to surface the small percentage of events worth human attention, though common automated attacks highlight common patterns to cover to keep an architecture secure.

6.2 Algorithm choice : Isolation Forest

We chose Isolation Forest [1] for the following reasons :

- Unsupervised learning : no labeled training data required. Since we have no ground truth labels ('this specific event was a targeted attack'), supervised algorithms like Random Forest cannot be applied without a pre-labeled dataset.
- Computation efficiency : Isolation Forest builds an ensemble of random decision trees and isolates points by randomly splitting features. Points that are easy to isolate (require few splits) are anomalies. This is extremely fast even on large datasets.
- No distribution assumptions : unlike statistical methods (z-score, Gaussian models), Isolation Forest makes no assumptions about the underlying data distribution.
- Directly interpretable output : the anomaly score (`decision_function`) provides a continuous confidence measure, not just a binary label.

6.3 Feature engineering

We used the following features to train the model:

Feature	Type	Description
dest_port	Numeric	Destination port of the connection
src_port	Numeric	Source port of the attacker
type_enc	Encoded categorical	Honeypot type (Honeytrap, Suricata, P0f...)
protocol_enc	Encoded categorical	Network protocol (TCP, UDP...)
ip_rep_enc	Encoded categorical	IP reputation (known attacker, mass scanner, unknown)
port_ratio	Derived numeric	$\text{dest_port} / (\text{src_port} + 1)$ — captures unusual port combinations

Table 7: Features used for the Isolation Forest anomaly detection model.

The VPS’ own IP address (172.252.236.68) was filtered from the dataset before training as Suricata logs outgoing connections from the server itself, which skewed the anomaly detection toward internal traffic rather than external attackers.

6.4 Model configuration

The model was configured with $\text{contamination}=0.05$, meaning we expected approximately 5% of connections to be anomalous. This parameter acts as a threshold on the anomaly score distribution — the bottom 5% of scores (most isolated points) are classified as anomalies. The choice of 5% reflects standard practice in network security anomaly detection: a lower value risks missing real threats, while a higher value produces too many false positives for analysts to review.

6.5 Results

The trained Isolation Forest classified the dataset into common automated attacks and anomalous events. Since the contamination parameter was set to 5%, the model marked the most statistically isolated connections as anomalies.

Classification	Count	Percentage
Normal (common automated attacks)	49,715	95.0%
Anomalies (unusual behaviour)	2,617	5.0%
Total	52,332	100%

Table 8: Isolation Forest classification results.

Anomalies by honeypot type:

Honeypot	Anomalies De- tected	Interpretation
Honeytrap	1,005	Unusual ports probed — unique scanning patterns
Suricata	697	IDS signatures on non-standard traffic flows
CiscoASA	470	All CiscoASA anomalies are “known attackers” — targeted Cisco exploitation
P0f	239	Unusual OS fingerprints — non-standard TCP stack implementations
ConPot	120	SCADA-specific attacks — relatively rare, high severity
Honeypots	86	Unusual service combinations targeted simultaneously

Table 9: Anomalies detected by honeypot type.

6.6 Heatmap analysis : Type x IP reputation

Cross-referencing anomalies with IP reputation revealed important patterns. CiscoASA anomalies came exclusively from IPs with established attack history (470 known attackers, 0 mass scanners), strongly suggesting targeted exploitation attempts against Cisco infrastructure rather than random scanning. Suricata detected the highest anomaly concentration with 527 events from known attackers on unusual ports, consistent with sophisticated actors trying to evade standard firewall rules. ConPot anomalies were split evenly between targeted actors and opportunistic scanners (52 and 51 events respectively), which is concerning from a critical infrastructure perspective given that ConPot simulates industrial SCADA systems. Honeytrap anomalies were dominated by mass scanners (239 events), consistent with infrastructure mapping tools like Shodan or Censys probing dynamic ports.

6.7 Top suspicious connections

The 10 connections with the lowest anomaly scores (most anomalous) after filtering the VPS’s own IP were dominated by IPs in the 167.94.146.x range — the Censys internet scanner — probing ephemeral ports (54000-65000 range) not associated with any standard service. This is consistent with Censys’s full-internet port scanning methodology, which systematically probes all 65,535 TCP ports.

7 Relevance to today's Network Security

7.1 Threat intelligence

Honeypots remain one of the most effective sources of ground truth threat intelligence. Unlike IDS signatures based on known attack patterns, honeypots capture zero-day techniques and novel scanning methodologies because any connection to a honeypot is inherently malicious — there is no legitimate reason for a system to connect to a honeypot. The IP addresses, ports, payloads, and timing data collected form high-quality IoC feeds with near-zero false positive rates.

7.2 AI integration in modern SOC's

The integration of machine learning into honeypot analysis addresses a fundamental scalability problem : at 30 million events per day, human review is impossible. Anomaly detection models like Isolation Forest enable automated triage — surfacing the small percentage of events that deviate from baseline behavior and routing them to human analysts. This *human-in-the-loop* model is now standard practice in enterprise SOC's.

More advanced applications include reinforcement learning based adaptive honeypots that modify their behavior based on attacker actions, and LLM-based honeypots (like Beelzebub, included in T-Pot's LLM mode) that generate contextually appropriate responses to fool attackers into extended interactions, yielding richer behavioral data.

7.3 Limitations

- Legal and ethical considerations : Exposing a honeypot creates legal grey areas in some jurisdictions. Operators must ensure they cannot be held liable if the honeypot is used as a stepping stone to attack third parties.
- Fingerprinting : Sophisticated attackers can detect honeypots through subtle behavioral differences (response timing, error message formatting, TCP stack anomalies). T-Pot partially mitigates this with high fidelity emulations, but it cannot be fully eliminated.
- Ressource cost ; The ELK stack required 8+ GB RAM, making honeynets impractical for resource-constrained environments without careful optimization.
- Data volume: 30 million events in two weeks required careful data management — bulk export hit Elasticsearch's query limits and required per-index pagination.

8 AI strategy

8.1 Overview

Our group used AI tools throughout this project, both for technical guidance and theoretical explanations. This section details how we use these tools, how we assessed the quality of the information provided, and what we learned about effective AI-assisted technical work.

8.2 Use cases

- Architecture decisions : We used it to help argue for the right installation, compare VPS providers and understand the tradeoffs between different honeypot types.
- Deployment troubleshooting : When we encountered some minor problems with Ansible (deploy), we described the error output and received immediate, accurate fixes and tips. This saved significant debugging time.
- ML model: It helped us find resources to help us understand the concepts behind Isolated Forest and Random Forest.
- It helped us explore and understand various theoretical and technical concepts throughout the project.

8.3 Correctness assessment

We did not blindly trust AI-generated information. Our verification approach:

- Cross-referencing with official documentation: All T-Pot configuration advice was verified against the official GitHub repository (github.com/telekom-security/tpotce). All scikit-learn parameters were verified against the official API documentation.
- Empirical validation: We ran commands and observed results before proceeding.
- Peer review: When one group member received AI guidance, they shared it with the group for review before implementation.
- Recognizing AI limitations: It occasionally provided outdated information or general advice that needed adaptation to our specific environment. We learned to treat AI output as a starting point, not a final answer.

8.4 Prompting optimisation

We found that providing full context significantly improved response quality. Providing, for example, actual error message and the specific index structure we observed yielded precise, actionable

solutions immediately. “Context - Request - Expected Response - Verification” is a proven process.

8.5 Individual reflections

9 Conclusion

This project demonstrated the full lifecycle of a real-world honeynet deployment and AI-assisted threat analysis. Starting from a theoretical understanding of deception-based defence, we deployed a production-grade honeynet infrastructure on a live VPS, collected over 30 million real attack events in two weeks, and built a functioning anomaly detection model on the collected data.

The key technical contributions of this project are: a working T-Pot Mini deployment with documented configuration decisions and lessons learned; a data extraction and preparation workflow querying Elasticsearch day-by-day, filtering noise, and preparing features for machine learning; and an Isolation Forest model trained on real honeypot data that detects 2,617 anomalous connections from 52,332 events, including targeted Cisco exploitation attempts and unusual SCADA attack patterns.

From a broader perspective, this project confirmed that honeypots remain highly relevant in modern network security architectures. The volume and diversity of attacks we observed — including SCADA-targeted traffic, medical system probes, and Cisco-specific exploitation — demonstrate that internet-exposed infrastructure faces constant, automated, and sophisticated threats. The integration of ML-based anomaly detection addresses the fundamental scalability challenge of honeypot data analysis and is representative of current SOC practices.

Future work could include geolocation enrichment of attacker IPs, temporal analysis of attack patterns across the collection period, comparison with the CICIDS2017 benchmark dataset, and testing of alternative algorithms such as DBSCAN or autoencoders against our data.

References

- [1] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, 2008.
- [2] Deutsche Telekom Security. T-Pot: A Multi-Honeypot Platform. GitHub, 2024.
- [3] MITRE Corporation. MITRE ENGAGE — A Framework for Cyber Deception.
- [4] Scikit-learn Developers. IsolationForest API Documentation.
- [5] Lance Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 2002.
- [6] Niels Provos and Thorsten Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 2007.
- [7] Censys Inc. Censys Internet-Wide Scanning Infrastructure.