Development Individual Project: Presentation Transcript

(Intelligent Agents Module)

Student Name: Koulthoum Flamerzi

Student ID: 32768

Slide 1: Title – Digital Forensics Agent System (DFAS)

Good [morning/afternoon] everyone and thank you for being here today. What I'll be sharing with you is the outcome of a team project where we developed the Digital Forensics Agent System, or simply DFAS.

Now, the main idea behind DFAS is to support digital investigators and analysts in their work—specifically, in collecting, processing, and preserving digital evidence in a way that's both reliable and efficient. If you've ever seen what happens after a cyber incident, you'll know investigators deal with huge amounts of data—everything from documents and spreadsheets to emails, images, and system files. Doing all of this manually can be slow, and it increases the chance of mistakes or overlooked evidence.

That's where DFAS comes in. Instead of replacing investigators, it helps them by automating the repetitive and technical parts of the process. This way, investigators can focus on analysis and decision-making, while the system ensures the evidence is gathered in a structured manner—kept intact, and ready to stand up in court if needed.

Slide 2: Overview & Objectives

Let's start with a quick overview of what we set out to achieve with DFAS.

Our first goal was **automation**. Normally, an investigator has to spend hours digging through folders, checking files one by one, and writing down details. DFAS takes over that part of the job—it automatically scans, identifies, and records the files that matter. This not only saves time but also reduces the chance of human error.

The second goal was **compliance with forensic standards**. In this field, it's not enough just to collect evidence—you have to do it in a way that will hold up in court. That means following strict guidelines like ISO 27037, NIST SP 800-86, and the ACPO principles. We designed DFAS with these standards in mind, so investigators can be confident that the evidence they collect is solid and defensible during legal proceedings.

The third objective is **forensic-grade outputs**. Every file the system processes is hashed with **SHA-256** to guarantee integrity, then securely packaged using **AES-GCM encryption**, and logged in a tamper-evident manner. This ensures we can always prove the evidence has not been altered.

To put it simply, DFAS is about creating an **automated, standards-driven, and trustworthy workflow** for digital evidence.

Slide 3: System Requirements

So how does one run DFAS?

The good news is that it doesn't require heavy infrastructure. It can run on **Windows, Ubuntu, or macOS**. The whole system is written in **Python 3.11**, which is widely used in the digital forensics and incident response community because of its portability and library support.

Speaking of libraries, DFAS uses several important ones:

- **watchdog** for monitoring file system events,
- **yara-python** for running YARA scans to detect patterns such as malware signatures,
- **python-magic** for file type identification,
- **cryptography** for encryption,
- and **SQLite** for storing evidence records in a local database.

We also left the door open for optional integrations with well-known tools like **Sleuth Kit** and **Autopsy**, though they're not necessary for the system to function.

In short, the system requirements are deliberately lightweight so DFAS can run on a typical investigator's workstation.

Slide 4: Agent Architecture

The heart of DFAS is its **agent-based architecture**, inspired by the **Belief-Desire-Intention (BDI) model**. Let me explain what that means.

In the BDI model:

- **Beliefs** represent the facts the system knows at a given moment.
- **Desires** are the goals the system wants to achieve.
- **Intentions** are the concrete plans or actions it commits to in order to achieve those goals.

This model is great for systems that need to make decisions in changing environments — exactly what we need for digital forensics.

In DFAS, we have four key agents:

1. **Discovery Agent** – responsible for scanning the file system and identifying candidate files.
2. **Processing Agent** – takes those candidates, calculates SHA-256 hashes, extracts metadata, and applies optional YARA scans.
3. **Packaging Agent** – builds encrypted evidence packages, generates CSV and JSON reports, and records chain-of-custody entries.
4. **Orchestrator Agent** – coordinates the workflow and ensures everything runs in sequence according to the policy.

Originally, our design also proposed a **Transport Agent** for uploading packages to a secure server or cloud bucket. However, in this implementation, we focused on local evidence handling, which is both simpler and easier to demonstrate in our context.

This modular setup allows DFAS to be reliable, scalable, and easy to extend.

Slide 5: Processing Pipeline

Let me now walk you through the **processing pipeline**, step by step.

1. **Plan** – The Orchestrator loads a policy. For example, we may specify that the system should look for .docx, .pdf, and .jpg files but ignore system directories. We can also set size limits or choose whether YARA scanning should be on or off.

2. **Discover** – The Discovery Agent goes through the file system, applying those filters, and pushes candidate files into a queue.
3. **Process** – The Processing Agent then takes each file from the queue, calculates its SHA-256 hash, extracts metadata such as size, timestamps, and ownership, and records everything in the evidence database.
4. **Package** – The Packaging Agent groups these records, generates CSV and JSON reports, creates an encrypted ZIP archive, and appends an entry to the chain-of-custody log.
5. **Present** – Finally, the outputs — the archive, database, and reports — are available for investigators to review directly or import into tools like Autopsy.

What's important here is the **flow and traceability**. At every stage, there's a clear record of what was done, by which agent, and when. That's exactly what you want in forensics: no ambiguity, no missing steps.

Slide 6: Security & Compliance

Security is the backbone of DFAS. Let me highlight the four principles we built into the system.

- **Immutability** – Original evidence is never modified. Files are always accessed in read-only mode.
- **Integrity** – Every file is hashed with SHA-256 at the source, and that hash is verified again at packaging time. If a file changes in between, it's immediately detected.
- **Confidentiality** – Evidence archives are encrypted with **AES-GCM**, which not only encrypts the data but also verifies its authenticity. That way, if someone tampers with the package, the system knows.
- **Auditability** – Every step is recorded in an **append-only log**. Entries are hash-chained so that if one record is altered, the chain breaks and tampering is obvious.

All of these choices tie directly back to international standards like ISO/IEC 27037 and ACPO principles. That's what makes DFAS not just functional but also **legally defensible**.

Slide 7: Implementation Evidence

Now, what does this look like in practice?

DFAS produces clear evidence of its work through logs and outputs. For instance, during a run you might see log entries such as:

- "Found 7 files" during the discovery phase.
- "Processed: DesignProposal.pdf (Hash: ffd64817…)" in the processing phase.
- "Evidence package created (Hash: 4f7ff546…)" once packaging is complete.

Alongside the logs, DFAS generates three important outputs:

- **CSV reports** for easy inspection in Excel.
- **JSON reports** for machine processing or integration with dashboards.
- The **SQLite evidence database**, which stores everything in a structured way.
- And finally, the **encrypted evidence package**, which bundles the reports and files securely.

All of this ensures that anyone reviewing the results can see exactly what was done, when, and by which agent. Evidence of these are pasted as screen short , in the next slide , as well we have already added a readme.md file so it is easy for us to test the code again

Slide 8: already talked about

Slide 9: Testing & Validation

We knew that a system like this would only be trusted if it was thoroughly tested.

We began with **unit tests** to check core functions like hashing, file type detection, and archive creation. Then we ran **integration tests** on a sample dataset. This dataset included normal files, files with spoofed extensions, large files, and even locked files to mimic real-world challenges.

We also tested for **repeatability**. Running DFAS twice on the same dataset should always produce the same hashes and outputs. That consistency is crucial for evidence to be admissible.

Finally, we validated the outputs. CSV and JSON were checked for correctness, the SQLite database was reviewed, and the encrypted package was verified against its hash. These steps give us confidence that DFAS is **robust, reliable, and ready for forensic use**.

Slide 10: Strengths & Challenges

Of course, no system is perfect. Let's look at both strengths and challenges.

Strengths:

- DFAS is standards-aligned, so its outputs are defensible in court.
- It's modular and agent-based, meaning components can be tested independently or scaled as needed.
- It's lightweight and portable, since it runs on Python with SQLite.

Challenges:

- File spoofing is still a problem. Someone could rename a .exe file to .pdf. To mitigate this, we use **libmagic** to check file headers, but even that has limits.
- Handling **large volumes of data** can slow the system down. We addressed this by adding filters, size limits, and multiprocessing, but scalability is still a challenge.
- Evidence security is critical. If someone were to gain access to stored archives, it would be a breach. We use AES-GCM encryption and key rotation to reduce that risk, but secure storage and key management always remain tough issues in practice.

Overall, the strengths outweigh the challenges, but acknowledging these limitations is important for future improvements.

Slide 11: Conclusion

To conclude, the **Digital Forensics Agent System** successfully meets its objectives. It can reliably identify, process, and preserve digital evidence in a way that is automated, standards-aligned, and auditable.

We've shown that it produces trustworthy outputs: encrypted archives, structured databases, and detailed reports. These outputs can then be used directly by analysts or imported into forensic tools for deeper investigation.

What makes DFAS particularly powerful is its **repeatability and transparency**. Every action is logged, every file is hashed, and every package is sealed with encryption. That means investigators and courts can have confidence in the results.

Looking forward, there are many ways this project could grow — for example, adding live monitoring features, integrating dashboards for visualization, or supporting larger databases for enterprise-level deployments. But even in its current form, DFAS is a practical tool for triage and first-pass evidence collection.

Thank you very much for your time and attention. I'll now be happy to take your questions.