BRABANT HACK_26

TRACK DEFENCE

DATA SCIENCE CENTER OF EXCELLENCE

WWW.BRABANTHACK.AI

AI-ASSISTED DECOMPILATION OF OBFUSCATED PYTHON BINARIES

Your mission: uncover what hidden code is really doing

CASE CONTEXT

In today's threat landscape, we see a clear shift in the way malware is being developed and distributed. Whereas traditional malware was almost exclusively written in languages such as C, C++, or assembly, recent years have shown a strong increase in Python-based malware, particularly in cloud environments, botnets, and ransomware. Python allows for rapid development, is platform-independent, offers a wide range of libraries, and can easily be packaged as a standalone executable using tools such as PyInstaller. For attackers, this means they can deliver advanced functionality with minimal effort while hiding code behind multiple layers of obfuscation and bytecode encoding.

The classic problem is that such binaries are difficult to analyze manually in practice, especially when attackers employ a variety of techniques to conceal code. Moreover, automated extraction tools often only result in fragmented reconstructions of the logic: variables lack meaningful names, control flow is distorted, strings are encrypted or compressed, and code is split into modular fragments that only come together at runtime.

Large Language Models (LLMs) offer new opportunities for code analysis, such as recognizing patterns in incomplete or syntactically irregular code. They can suggest meaningful variable names, reconstruct hidden semantics, and formulate hypotheses about the purpose of a function or code fragment. The question is: **how far does this LLM capability really extend?** Can AI help decompile obfuscated malware faster and more accurately? Can AI reconstruct behavior where traditional analysis methods fail? The ultimate goal is to accelerate malware analysis.

The importance of this case is twofold. Operationally, AI-driven decompilation can speed up incident response, allow faster identification of indicators of compromise, and enable timely detection of threats. Strategically, it contributes to the development of autonomous or semi-autonomous cybersecurity capabilities. This case is therefore both practical and societally relevant.
As a participant you will investigate how Large Language Models (LLMs) and AI-driven analysis techniques can be used to decompile obfuscated Python binaries and reconstruct their underlying behavior. The challenge mirrors a realistic scenario in which modern malware or droppers are delivered as PyInstaller-like executables that;

1. contain no visible .py source code,
2. apply multiple layers of obfuscation, and
3. dynamically load code via exec() or eval() and use control-flow flattening.

The goal is to explore to what extent AI models are capable of reconstructing code, interpreting it, and understanding its semantics, even when traditional reverse engineering techniques yield only fragmentary clues. This case is primarily research- and defense-oriented and is not intended as a pitch platform for commercial product development during the hackathon.

WHAT DO TEAMS BUILD?

You will build an AI-driven prototype demonstrating how an obfuscated Python binary can be automatically decompiled and analyzed. This may take the form of a working pipeline, dashboard, visualization, or a conceptual demonstrator (e.g., slides, a notebook, or a web interface) that clearly illustrates how the AI solution reconstructs, inspects, and interprets the binary's behavior. All teams will submit their code and documentation so that the solution can be reviewed after the hackathon and potentially further developed within the Defense organization.

WHAT DOES DATA SCIENCE CENTER OF EXCELLENCE PROVIDE?

We will provide a set of safe, synthetic Python binaries that are obfuscated and simulate ransomware-like behavior, including several variants with different levels of complexity. In addition, we will supply basic tools for extracting bytecode (such as a PyInstaller extractor), a starter notebook with example code for interacting with AI models, and a brief technical explanation on the day itself describing how the binaries are constructed. This enables you to get started immediately without having to develop malware or tooling themselves.

SUCCESS CRITERIA

The solution is considered successful when you and your team demonstrate that your AI prototype can automatically reconstruct the essence of an obfuscated Python binary's behavior and present it in a clear, reproducible, and interpretable manner.

Core KPI:
The degree to which the AI solution correctly identifies and describes the core behavior of the binary (e.g., encryption, persistence, network activity, or other malicious logic).
Boundary conditions:

1. Reproducibility: The pipeline, demonstration, or mock-up must consistently produce the same insights when executed repeatedly on the provided binaries.
2. Transparency of analysis: The output must be understandable to a technical audience (e.g., through explanations, visualizations, or workflows), clearly showing how the AI arrived at its reconstruction and which analysis methods were used.
3. **Transferability**: All produced code, notebooks, or prototypes will be made available to Defense, allowing the insights to be continued, tested, or integrated into follow-up research after the hackathon.

CONTACT

- Mark Patrick Roeling (mp.roeling@mindef.nl)

WWW.BRABANTHACK.AI