apriorit

CASE STUDY

From Vulnerable to Bulletproof: Securing Intellectual Property Through Reverse Engineering

REVERSE ENGINEERING PYTHON

Background

For companies developing specialized software, intellectual property protection is essential to maintain a competitive advantage. Our client, a heavy machinery software developer, needed to assess the vulnerability of their proprietary solutions.

Our security team conducted a comprehensive reverse engineering assessment, simulating real-world attack scenarios to demonstrate how malicious actors could extract the client's software logic. We then provided actionable recommendations to strengthen IP protection.

The client

Our client specializes in automation software for heavy industrial machinery. They create innovative technologies to make machinery more efficient and safer to operate. The client's software products are based on the company's intellectual property (IP) and patented solutions, so protecting data and IP is a top priority.

Client:
NDA

Location:
NDA

Industry:
Manufacturing automation

Collaboration with Apriorit:
One-time project

The challenge

The client was looking for a cybersecurity team that could assess the level of intellectual property protection for their DLL files. In particular, they wanted a team that could:

- Determine whether the source code could be easily restored from compiled file
- Accurately estimate the time, skills, and effort required to recover application logic
- Suggest ways of securing intellectual property and making logic recovery more challenging or impossible

The kind of research the client was looking for required significant reverse engineering expertise. Another challenge was the fact that the client's DLL files were written in Python and differed significantly in their structure from typical C++ compiled files, which made it impossible to use automated reversing and code analysis tools.

After assessing teams with deep reverse engineering skills, an understanding of Python, and cybersecurity expertise, the client decided to go with Apriorit.

Project details

Apriorit team

Project manager

Reverse engineers

Python software developer

Tech stack

LANGUAGES

C++

Python

TESTING TOOLS

IDA Pro

Ghidra

Looking for a reliable reverse engineering partner?

Apriorit will provide you with in-depth reverse engineering expertise and unique skills to help you protect your product and business.

CONTACT US →

The result

With the help of manual file analysis and reverse engineering, the Apriorit team recovered both file and software logic from the provided DLLs. We also found a way to automate our approach and showed the client practical ways in which a threat actor could steal IP from their software.

Additionally, we suggested more reliable obfuscation methods and a plan to secure their intellectual property.

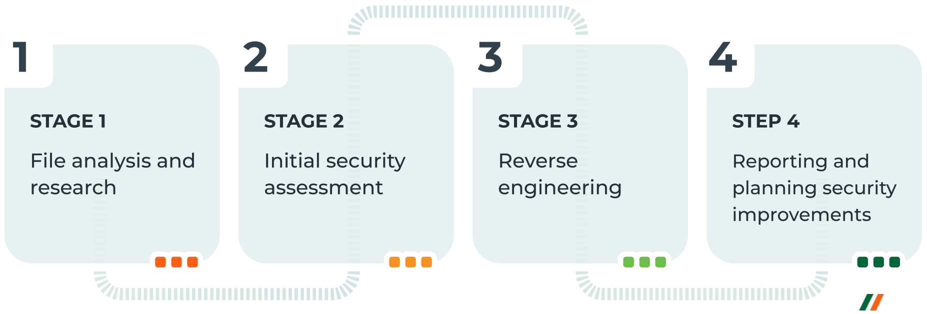
How we did it

After the initial project assessment, we gathered a team consisting of a project manager, a Python developer, and several reverse engineers. This team had extensive experience with Python, C++, x86-64 Assembly, IDA Pro, and Ghidra, which they used for file analysis.

The client wanted us not only to recover file logic from the DLLs they provided but also to research whether those DLLs provided deeper insights into their software, which is based on intellectual property. They also asked our team to estimate the time required to restore the original Python code across the entire project.

With this in mind, we divided the project into four key stages:

4 stages of investigating IP protection


1
STAGE 1
File analysis and research
2
STAGE 2
Initial security assessment
3
STAGE 3
Reverse engineering
4
STEP 4
Reporting and planning security improvements

Stage 1. File analysis and research

The client provided our team with several DLL files that contained IP from their real project. Initial analysis of those files helped us to:

- Identify the file format and target platform
- Extract useful strings
- Detect that the client's team used the Cython library to compile Python code

Cython is a tool that compiles Python code into C extensions, boosting the performance of interpreted Python code to the level of C code. Using Cython also helped the client somewhat protect their intellectual property from reverse engineering, as automated reverse engineering tools are geared towards files written in C++. Even Apriorit's veteran reverse engineers had to research ways to restore Python code from the Cython compiled file, as this is an exotic option that is not commonly used.

We discovered that it was impossible to restore the code with existing tools for automated code recovery, as they need at least Python bytecode or pieces of the source code. However, the client's IP could still be vulnerable to manual recovery and analysis, so our team geared up for deeper cybersecurity research.

Stage 2. Initial security assessment

At this stage, we evaluated the complexity of reverse engineering the code by analyzing its structure and the APIs utilized. We tried using traditional reverse engineering tools like IDA and Ghidra, but they provided inaccurate or misleading information in their decompilation interfaces.

At this point, we estimated that recovering the code and software logic would be challenging, making the software unlikely to attract an attacker. However, the client wanted to know if it was possible, so we moved to manual reversing and code recovery.

Stage 3. Reverse engineering the files

During in-depth reverse engineering, the Apriorit team managed to:

- Find a way to restore strings and constants from compiled code
- Write a Python script to automate recovery, which allowed us to accelerate the process and recover multiple files simultaneously
- Identify code construction patterns and key points that indicate correlations between lines of Python code and sets of instructions in the compiled file
- Restore the main part of the Python code and showed the client practical ways in which a threat actor could steal IP from their software. the original version

Once we reverse engineered the provided files, we understood the structure of the .pyd file, detected important components, and developed a method to recover the code. At this stage, our team designed a proven and reliable way to recover application logic and put the client's IP in danger. Now, we needed to propose a solution to this issue.

Stage 4. Reporting and planning security improvements

We provided the client with samples of recovered code to confirm the success of our research, as well as a report on our activity. The report included our assessment methodology, steps we took during reverse engineering, and findings.

Once the client confirmed our findings with their development team, we started working on the final part of the project — helping the client protect their intellectual property.

Drawing on over 20 years of cybersecurity expertise, we offered our client several ways to obfuscate their software code so that it remains safe. We explained the pros and cons of each option, the difficulty of implementation, and possible impacts on the software. The client chose one of the options and started discussing a new obfuscation project with our team.

Impact

The Apriorit team helped the client understand security issues in their software and vulnerabilities that may cause IP leaks or theft. We provided an assessment of the software's current security level and an automated pipeline to reverse engineer files.

Based on our research, we offered ways to improve the client's software security with more reliable obfuscation algorithms. The client already plans to work with us to help them navigate challenges in securing intellectual property.

Protect your intellectual property before it's jeopardized

Leverage Apriorit's expertise in reverse engineering and security-focused development to protect your code.

CONTACT US →