

Internship offer (Master 2)

Security-Oriented Differential Analysis of Binary Updates

Supervisors Yann Thierry-Mieg, [Dylan Marinho](#) 
Yann.Thierry-Mieg@lip6.fr, Dylan.Marinho@lip6.fr
Laboratory LIP6, CNRS UMR 7606, Sorbonne Université
Team MoVe (Modeling and Verification)

Keywords software supply chain security · binary analysis · binary diffing · reverse engineering · vulnerability detection · program analysis

1. General Context

In many operational contexts, the binaries that are actually deployed are the only artifacts consistently available for analysis, auditing and compliance verification. Attackers exploit this systematically: by compromising third-party components, build pipelines or distribution channels, an adversary can inject malicious or vulnerable code that reaches production systems without ever appearing in the source repository [GAH26].

The 2024 XZ Utils backdoor¹ is the defining recent example. The payload was not introduced in the visible source code, but through a test file invoked by the autoconf build system, and the resulting binary would have exposed SSH daemons to unauthenticated remote code execution. The attack succeeded because every traditional integrity check passed: the repository signature was valid, the source code looked clean, and the project had a credible multi-year history. The same holds for the SolarWinds (2020) and 3CX (2023) compromises: in each case, the binary was the last—and only—layer where the modification was objectively present.

2. Scientific Context

Given two versions of a binary, we would like to answer a simple question: what is different, and is any of it dangerous? Today, two families of tools exist and they only answer half of it.

Binary diffing tools² align control-flow graphs and disassembled code to highlight structural differences. They remain descriptive: they report that something changed, not whether the change

¹CVE-2024-3094, rated 10.0 on the CVSS (*Common Vulnerability Scoring System*) severity scale.

²e.g. BinDiff [Goo23], Diaphora [Dia24] or ghidriff [Cle25], built on top of reverse-engineering frameworks such as Ghidra [Nat26]

matters for security. They are moreover sensitive to compilation noise, since a compiler upgrade or a different inlining decision produces large differences with no semantic content.

Conversely, capability and vulnerability detection operates on a single artifact [AK25], either by signature matching (YARA [Vir25], capa [Man25]) or through heavier semantic analyses (angr [SWS+16], BinSec [DB15]). Such tools cannot distinguish a newly introduced risk from one that was already present—and already accepted—in the previous version.

A few recent works bridge the two sides, by summarising diffs with large language models [UPK+25] or by comparing the capabilities detected in both binaries [Cha26], but the question of the security impact of a change remains largely open.

3. Objectives

The goal of this internship is to explore, experimentally, how far one can get by combining the two families of tools above, and where the difficulty really lies. The work is organised along three directions, which can be adjusted depending on the interests of the candidate.

- **Take existing tools in hand.** Apply diffing and capability-detection tools to pairs of binaries—documented cases such as the XZ backdoor, or CVE-linked before/after pairs—and characterise what each of them can, and cannot, express about a change.
- **Represent the differences.** Propose a first structured representation of a *delta* between two binaries, recording the changed zones together with annotations produced by the analyses.
- **Assemble and experiment.** Connect these building blocks into a simple chain, from a pair of binaries to a readable report, and try it on a small set of examples.

The expected outcome is a prototype, even partial, and an experimental report. The limits encountered are part of the result: understanding *why* a simple pipeline fails to separate benign recompilation noise from meaningful change is already a useful contribution.

4. Skills

The internship is open to Master 2 students in computer science, with:

- programming skills (e.g. Python, C, C++, Java);
- interest in computer security, reverse engineering or low-level software;
- notions of compilation and of executable formats.

5. Conditions

The internship will take place at LIP6 in the MoVe (Modeling and Verification) team, within Sorbonne Université in Paris.

The LIP6 is a joint research unit (UMR 7606) of Sorbonne University and the CNRS. It is a computer science research laboratory with approximately 450 members, including over 170 permanent researchers.

The BINTRACE project. The internship is part of the BINTRACE project, which brings together the MoVe team at LIP6, the LORIA (Université de Lorraine, Carbone team) and the industrial partner S3NS³. BINTRACE aims to validate binary updates incrementally, starting from a trusted first version and establishing a certification path up to the current one, without relying on *trust* in the provider. The intern will therefore have the opportunity to interact with the project partners: the LORIA, which brings expertise in binary-level semantics, and the security analysts of S3NS, who work on real update streams.

PhD position. If the internship is successful, it may be pursued by a PhD position on the same topic with the same supervisors.

Duration. 5 to 6 months, starting in February or March 2027, with the standard internship stipend.

Application. By email first, with a CV and recent transcripts.

References

- [AK25] Ashish Adhikari and Prasad A. Kulkarni. “Survey of techniques to detect common weaknesses in program binaries”. In: *Cyber Security and Applications* 3 (2025), page 100061. DOI: 10.1016/J.CSA.2024.100061.
- [Cha26] Chainguard. *malcontent*. 20 March 2026. URL: <https://github.com/chainguard-dev/malcontent/releases/tag/v1.21.5>.
- [Cle25] ClearBlueJar. *Ghidriff: Ghidra Binary Diffing Engine*. 4 July 2025. URL: <https://github.com/clearbluejar/ghidriff/releases/tag/v1.0.0>.
- [Dia24] Diaphora. *Diaphora: A Free and Open-source Binary Diffing Tool*. 30 May 2024. URL: <https://github.com/joxeankoret/diaphora>.
- [DB15] Adel Djoudi and Sébastien Bardin. “BINSEC: Binary Code Analysis with Low-Level Regions”. In: *TACAS 2015*. Edited by Christel Baier and Cesare Tinelli. Volume 9035. Lecture Notes in Computer Science. London, UK: Springer, 2015. pages 212–217. DOI: 10.1007/978-3-662-46681-0_17.
- [GAH26] Betul Gokkaya, Leonardo Aniello, and Basel Halak. “Software supply chain: A taxonomy of attacks, mitigations and risk assessment strategies”. In: *Journal of Information Security and Applications* 97 (2026), page 104324. DOI: 10.1016/J.JISA.2025.104324.
- [Goo23] Google. *BinDiff*. BinDiff is an open-source comparison tool for binary files to quickly find differences and similarities in disassembled code. 25 September 2023. URL: <https://github.com/google/bindiff/releases/tag/v8>.

³S3NS, a joint venture between Thales and Google, operates a SecNumCloud-certified sovereign cloud.

- [Man25] Mandiant. *capa*. 20 November 2025. URL: <https://github.com/mandiant/capa/releases/tag/v9.3.1>.
- [Nat26] National Security Agency (NSA). *Ghidra Software Reverse Engineering Framework*. 4 March 2026. URL: https://github.com/NationalSecurityAgency/ghidra/releases/tag/Ghidra_12.0.4_build.
- [SWS+16] Yan Shoshitaishvili, Ruoyu Wang, Christopher Salls, Nick Stephens, Mario Polino, Andrew Dutcher, John Grosen, Siji Feng, Christophe Hauser, Christopher Krügel, and Giovanni Vigna. “SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis”. In: *SP 2016*. San Jose, CA, USA: IEEE Computer Society, 2016. pages 138–157. DOI: [10.1109/SP.2016.17](https://doi.org/10.1109/SP.2016.17).
- [UPK+25] Meet Udeshi, Venkata Sai Charan Putrevu, Prashanth Krishnamurthy, Prashant Anantharaman, Sean Carrick, Ramesh Karri, and Farshad Khorrami. “Binary Diff Summarization using Large Language Models”. In: *arXiv (preprint)* abs/2509.23970 (2025), Submitted on 28 Sep 2025. DOI: [10.48550/ARXIV.2509.23970](https://doi.org/10.48550/ARXIV.2509.23970). arXiv: 2509.23970.
- [Vir25] VirusTotal. *YARA*. 30 October 2025. URL: <https://github.com/VirusTotal/yara/releases/tag/v4.5.5>.