**angr: A Powerful and User-friendly Binary Analysis Platform**

Yan Shoshitaishvili  
yans@asu.edu  
Arizona State University  
U.S.A.

Ruoyu Wang  
fishw@asu.edu  
Arizona State University  
U.S.A.

Audrey Dutcher  
dutcher@asu.edu  
Arizona State University  
U.S.A.

Christopher Kruegel  
chris@cs.ucsb.edu  
University of California, Santa Barbara  
U.S.A.

Giovanni Vigna  
vigna@cs.ucsb.edu  
University of California, Santa Barbara  
U.S.A.

**ABSTRACT**

angr is an open-source binary analysis platform that was released as the research artifact of a paper published at 2016 IEEE Symposium on Security & Privacy. Over the past eight years, angr has made significant impact in academia, industry, government, and among the enthusiasts of binary analysis. This short paper provides basic information about angr, its ecosystem, and its impact.

**KEYWORDS**

Binary Analysis, Vulnerability Discovery

**ACM Reference Format:**


1 INTRODUCTION

angr is an open-source binary (and program) analysis platform that was released under the BSD-2 license as the artifact to the “angr paper” [58] published at 2016 IEEE Symposium on Security & Privacy. The main project is on GitHub at https://github.com/angr/angr, with about 6.8k "stars" and over 1,000 forks at the time of writing. We wrote the first line of angr code in August 2013, and the year of 2023 marks the 10th birthday of angr.

Beneath angr are several key libraries that have been independently and widely used in the security community, including:

- **ClE**, a Pythonic generic binary loader.
- **PyVEX**, a library for interfacing with libVEX, which is the lifter and translator that Valgrind uses.
- **Claripy**, a Python library that provides arithmetic and symbolic abstraction for angr.
- **archinfo**, a Python library that provides architecture abstraction.

The main angr project, all these underpinning libraries, as well as a (mostly) functioning graphical user interface (GUI) angr management, are hosted under the angr organization on GitHub at https://github.com/angr. The documentation of angr can be found at https://docs.angr.io.

At the time of writing, excluding blank lines and comment lines, angr, angr management, and angr’s libraries contain about 551k lines of Python code, 44k lines of C code, and 3.5k lines of C++ code.

1.1 Creating an Open and Sustainable Architecture

We pride ourselves with the creation of angr’s open and sustainable architecture. A research publication, or even a source code release, is not sufficient to create an open and sustainable architecture. angr became a favorable choice of binary analysis research and engineering through a concerted effort.

**Developing in the open.** Before angr’s release, a common paradigm was the Research-Publish-Push cycle: Groups would only push updates to their prototypes when the paper using that code was published. This heavily discouraged the use of both FuzzBall [10, 59] and the Binary Analysis Platform (BAP) [6, 13], two older binary analysis frameworks that failed to achieve traction before angr’s release. First, by using such a prototype of another research group, one had to accept the fact that they were forced into a disadvantage: The latest and most powerful version of the prototype was, by definition, not the one they were using. Second, any bug fixes or improvements they made to the prototype would likely have to be discarded due to merge conflicts when the new version was pushed. Conversely, angr is developed in the open, with research project changes immediately upstreamed, and the only releases delayed are add-on modules built for specific projects. This maintains the community’s trust that they are using the best architecture that we can provide, and that contributions will be relevant.

**Creating an active community.** angr thrives, in a large part, because of an active community of researchers, engineers, hackers, and enthusiasts that frequent our GitHub issues pages and our public channels to offer help. Discussions,
issues, and pull requests occur on a daily basis. This improves angr’s usability, teaches new users and contributors to use and contribute to angr, and gives researchers confidence that angr is something that can actually be used.

**Interoperability with existing tools.** angr interacts with and uses existing tools wherever possible. There are bindings between angr and IDA Pro, angr and Binary Ninja, and angr and radare2. Likewise, angr depends on standard components of the ecosystem: Unicorn Engine is used for concrete execution, ELFTools for ELF file loading, and so on. This allows the architecture to benefit from parallel advancements by underlying libraries.

**Active maintenance and development.** Since its public release in 2015, the angr team kept improving angr by adding new features, improving its runtime performance, refactoring its codebase, and maintaining published research artifacts. To this end, the angr team invested a significant amount of engineering and innovation effort. Some examples include, the migration from Python 2 to Python 3 (in 2017), the introduction of angr decompiler (open and continuous development since summer 2017), the creation of angr IL (AIL), the creation of program interaction description library (archr [61]), the introduction of type hints (since 2022), and the maintenance of key components in the Mechanical Phish that rely on angr (angrop [57] and Rex [62]). While many other binary analysis platforms (especially research-facing ones) are gradually dying, angr is still extremely active.

# 2 IMPACT

Since we released angr to the public in August 2015, it has made significant impact in the research, educational, industry, and government communities.

## 2.1 Academic Impact

On the academic side, the reference “angr paper” [58] has been cited 1,100 times, and many researchers simply cite the angr website [60] instead. The framework itself is actually used (i.e., not just cited as a related work) as a base by a significant body of work by researchers unaffiliated with the authors of the framework. We list some key research areas and papers below:

- angr is used as a base for vulnerability detection techniques in userspace binaries [17, 28, 29, 39, 77, 78];
- angr is used in the firmware of a wide variety of devices, from USB peripherals [31] to Industrial Control Systems [38], and beyond [23, 25, 44, 49];
- angr is used to detect potentially vulnerable code clones [5, 33, 47, 75] and patches [76];
- angr is used in advancements in automatic exploit generation techniques, with high-profile targets including secure enclaves [16], OS kernels [20, 68, 72, 73], and traditional user-space binaries [43, 67];
- angr is used in the development of novel code reuse attacks [9, 36, 42, 69];
- angr is used to create novel vulnerability mitigation techniques [1, 15, 22, 35, 50, 55, 75];
- angr is used to build automatic repair of binaries [32] and running systems [19];
- angr is used in code optimization [8];
- angr is used in code obfuscation [74] and deobfuscation [37, 40, 63];
- angr is used in binary code deboating [12];
- angr is used in binary attribution [2], binary code similarity detection [34, 66], and watermarking [45];
- angr is used to analyze malware targeting various operating systems and device classes [4, 7, 14, 27, 56];
- angr is used to reason about the security of hardware architectures [11, 64];
- angr is used in protocol analysis [21];
- angr is used in the creation of security visualizations [3];
- angr is used in a number of other developments in static analysis [18, 24, 30, 41, 46, 51, 54, 70, 71];
- angr has also been utilized as a component by composite program analysis, vulnerability detection, and hacking tools [48, 52, 53].

From our analysis of recent publication trends, angr appears to be used in roughly one fifth of all binary analysis research in academia.

## 2.2 Educational Impact

Aside from its use as a research tool in academia, angr is also heavily used in educational cybersecurity competitions. Upon its release, it “raised the bar” of cybersecurity challenge difficulty with its ability to automatically solve many reverse engineering problems [26]. For example, the year after angr’s release, the organizers of DEF CON CTF (the world championship of cybersecurity competitions) fielded fewer reverse engineering challenges than other types of challenges in their qualifying event. Anecdotally, from conversation with the organizers, this seemed to be due to the difficulty in making “angr-proof” challenges. Since then, competition organizers have learned to create increasingly harder challenges to stump angr, further challenging participants and increasing the value that they can get from such events.

In a further example of angr’s impact on education, in the 2019 CSAW Embedded Security Challenge hardware hacking competition, a challenge focused on reverse engineering and exploitation of embedded devices, nine of the top 10 teams used angr for their solutions.

## 2.3 Industrial Impact

angr has also impacted industry. It is used by a wide range of companies across different market segments, from Apple and Samsung to Boeing and Raytheon, and a number of national labs, FFRDCs, and similar entities.

## 2.4 Governmental Impact

angr has impacted government, with the angr-based Arbiter [65] analysis technique utilized by cyber operators in the Department of Defense to help accomplish mission goals.
REFERENCES


