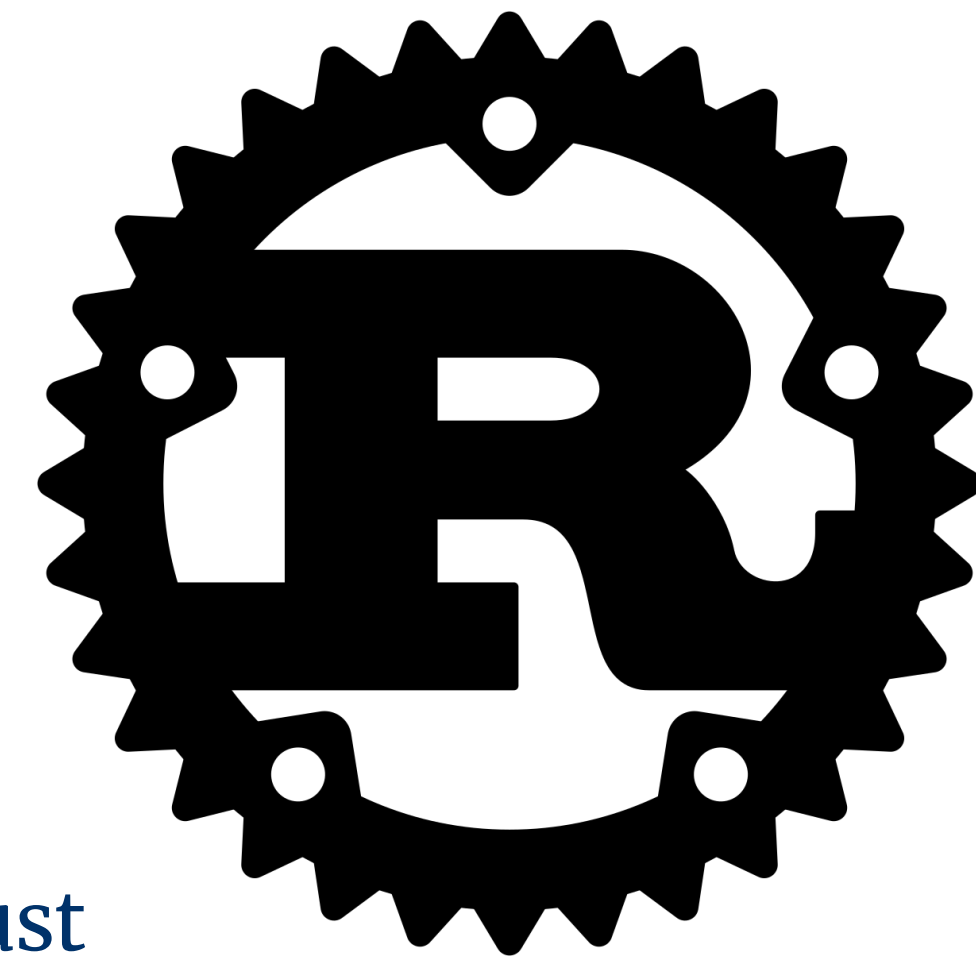


Cargo Sherlock: A Formal Analyzer for Supply Chain Trust

Muhammad Hassnain, Anirudh Basu, Ethan Ng, Caleb Stanford

INTRODUCTION

- Formal model for *trust* in the software supply chain that incorporates human factors
 - Built on solving for the *minimum trust* needed to believe that a library is safe
 - Uses a first-order relational model and a SMT solver to solve the minimum trust problem
- Cargo Sherlock is an automated reasoning tool for modeling trust in the Rust software supply ecosystem



GOALS

- Compositional:** Trust costs for different portions of the package ecosystem should be able to be computed independently, and then combined.
- Auditable:** A trust cost should correspond to real properties and/or factors that were considered about the software, that are human-readable.
- Specific:** The analysis is committed to a specific set of assumptions, which allow one to conclude that the assumption(s) made were false if the code is later found to violate trust.
- Configurable:** The user should be able to add components into the tool, including the results of additional program analysis tools or other human factors.



USER INTERACTION MODEL

In a time before Cargo Sherlock...

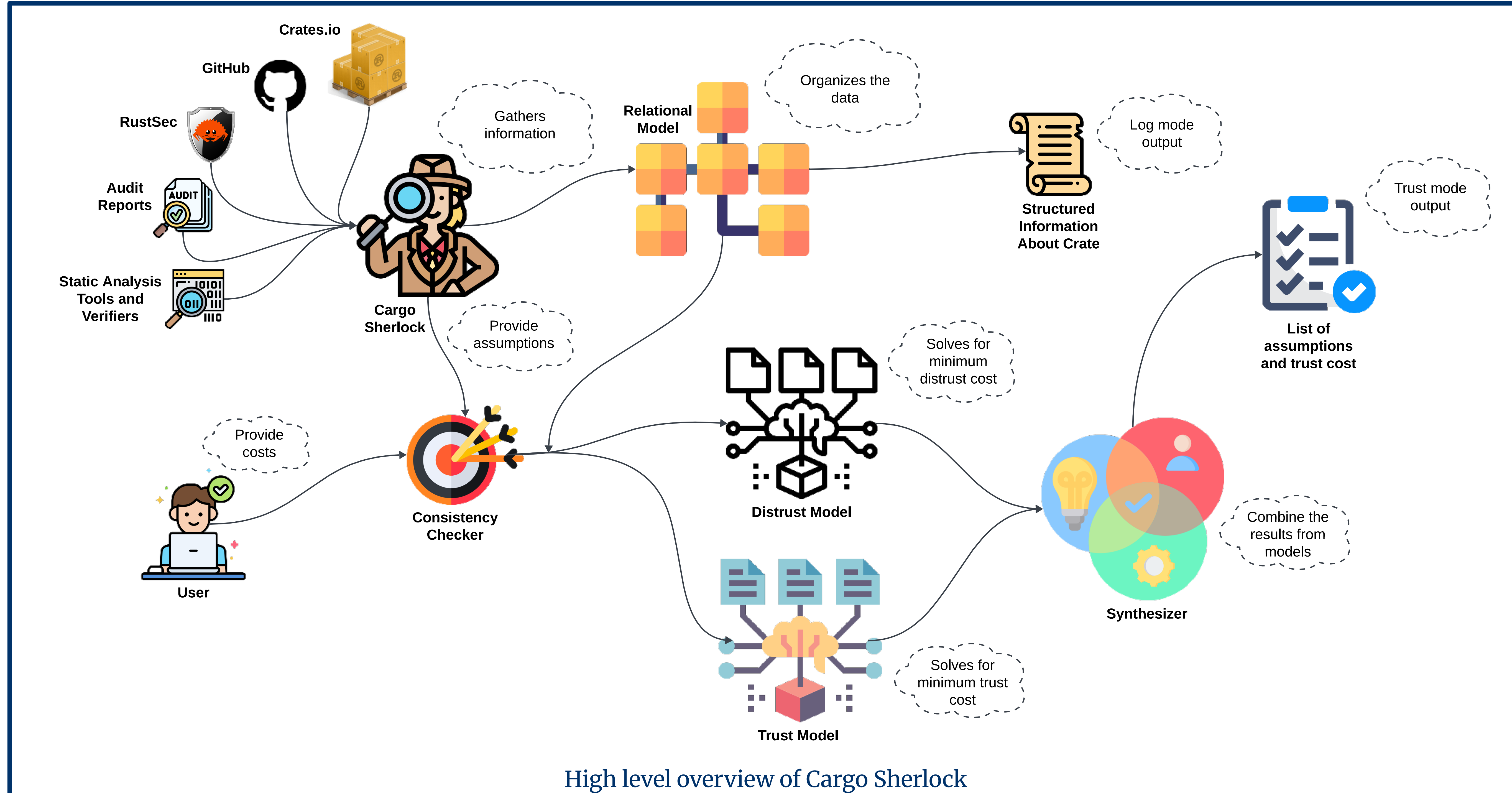
- Ferris wants to download a crate from crates.io called `aho_corasick`
- Ferris checks and verifies all the API calls exist on the documentation
- Ferris might then go to the terminal and just write `cargo add aho_corasick`
- Ferris' program might be infected with malware because `aho_corasick` is malware!

With Cargo Sherlock (Repeat steps 1 and 2)

- Ferris asks Rust Holmes Sherlock (Cargo Sherlock) whether or not the crate is safe
- If it returns **safe** or **low severity**, then we can trust the crate!

Otherwise, Ferris should investigate further before using it.

- * For more advanced users, they can pre-configure different costs and different assumptions depending on their particular use case!



MINIMUM TRUST PROBLEM

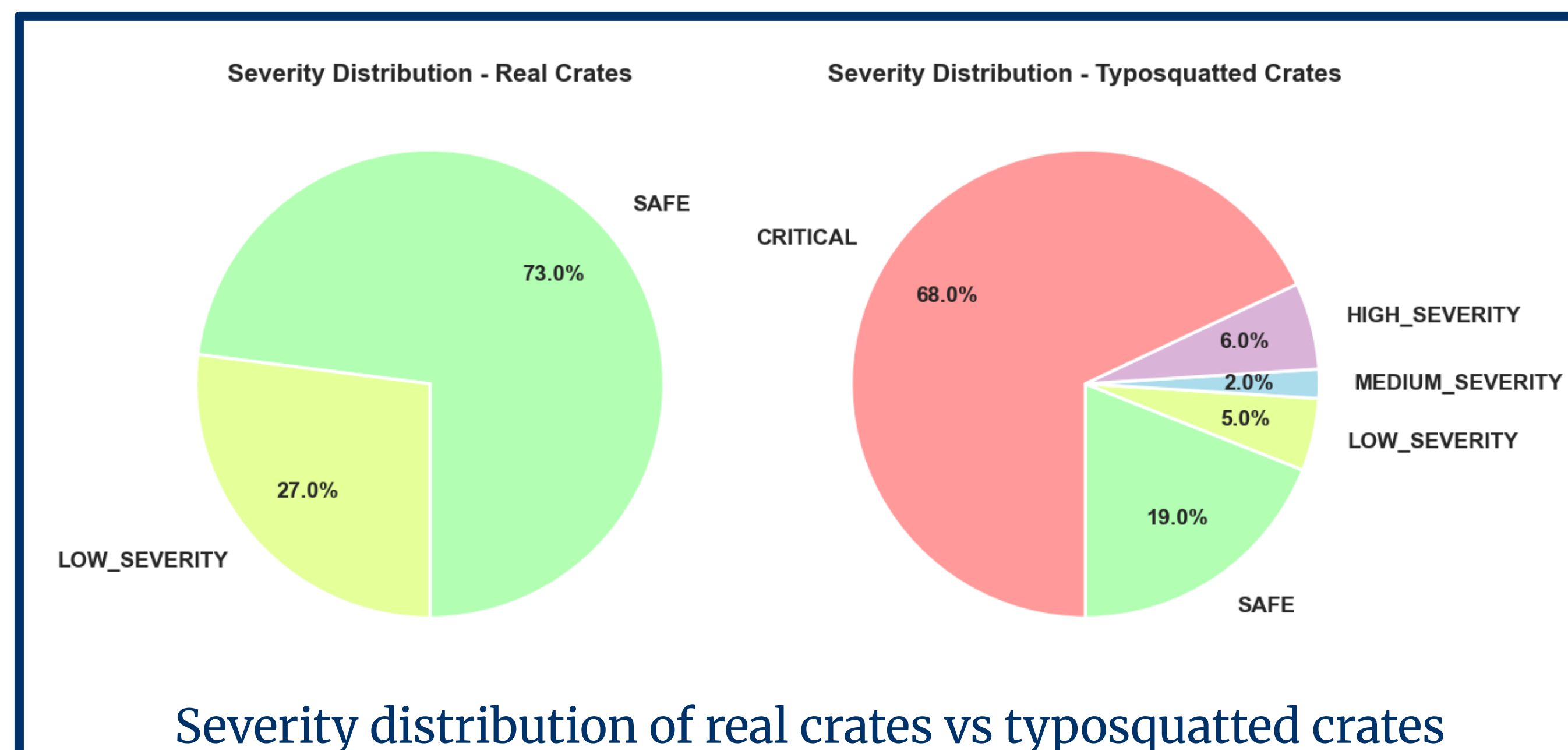
- The *minimum trust problem* attempts to find the *most reasonable valid* argument for some conclusion.
- The input to the minimum trust problem is a set of candidate assumptions in propositional logic, each with associated *trust costs*, and a conclusion.
- Trust costs represent how much it costs to make the assumption; a higher cost means a more unfounded assumption.
- The output is the set of assumptions with minimum cost that *prove* the conclusion.
- The output corresponds to the most reasonable valid argument.
- Cargo Sherlock automates solving the minimum trust problem using the Z3 theorem prover.

MINIMUM TRUST EXAMPLE

- In the example to the right, the goal is to prove that `aho_corasick` is safe by making assumptions with minimum cost.
- One way to prove `aho_corasick` is safe is by making the assumption number 1. However, this incurs a cost of 100!
- A better solution is to make assumptions 3 and 4. Together, these prove that `aho_corasick` is safe with a cost of 50.
- The best solution is to make assumptions 5, 6, 7, and 8. These four assumptions prove that `aho_corasick` is safe with a cost of 45!

FINDINGS

- ✓ We detect synthetically constructed supply chain attacks
- ✓ Cargo Sherlock detects real world supply chain risks
- ✓ Comparable results with existing database of safe and unsafe Rust crates from human curation



No.	Assumption	Cost	Encoding
1	aho_corasick is safe.	100	c
2	memchr is safe.	100	m
3	aho_corasick has many downloads.	30	d
4	If aho_corasick has many downloads, then it is safe.	20	$d \implies c$
5	Alice is trustworthy.	20	a
6	Bob is trustworthy.	5	b
7	If Alice is trustworthy and memchr is safe (aho_corasick's dependency), then aho_corasick is safe.	10	$(a \wedge m) \implies c$
8	If Bob is trustworthy, then memchr is safe.	10	$b \implies m$

Candidate assumptions for crate `aho_corasick`