**🔧 Tool Name: Ghidra + goresym**

**📚 History:**

* **Ghidra** was released by the NSA in 2019 as a powerful, open-source reverse engineering framework.
* **goresym** is a tool created to assist with reverse engineering Go binaries by recovering stripped symbols.
* The combination emerged from the need to better analyze Go binaries, which are often stripped and hard to interpret in tools like Ghidra or IDA Pro.

**📝 Description:**

This integration combines **Ghidra** for disassembly and analysis with **goresym**, which extracts and demangles Go symbol names from binaries. When used together, they enable meaningful static analysis of stripped or obfuscated Go executables.

**❓ What Is This Tool About?**

Ghidra by itself struggles with Go binaries, especially when symbols are stripped. goresym extracts function names and other debug information from Go binaries using Go's internal symbol table structure. These symbols can then be imported into Ghidra to label functions and variables—greatly enhancing understanding of the binary.

**⭐ Key Characteristics / Features:**

* Parses Go symbol tables (.gopclntab, etc.)
* Restores original function names in stripped Go binaries
* Outputs symbols in a usable format for further analysis
* Can be scripted to feed into Ghidra
* Supports a wide range of Go versions (especially 1.2–1.18+)

**🧩 Types / Modules Available:**

* **goresym CLI tool**: Run on binaries to extract symbols
* **Output formats**: Address-to-symbol mappings in plaintext or JSON
* **Ghidra import scripts** (custom): Python or Java-based loaders for applying names in Ghidra
* (Optional) **Batch rename scripts** in Ghidra to automate labeling

**🛠️ How Will This Tool Help?**

* Saves reverse engineers time by recovering function names
* Makes static analysis of Go binaries more productive
* Improves understanding of code flow, especially for malware or closed-source apps
* Helps identify key routines that would otherwise be marked as FUN\_xxxxxxxx

**🖼️ Proof of Concept (PoC) Images:**

*(You would insert before-and-after screenshots from Ghidra here:)*

* **Before**: All functions named FUN\_0xXXXXXX
* **After**: Functions named like main.main, runtime.mallocgc, net/http.(\*Server)

**📋 15-Liner Summary:**

Ghidra is a strong reverse engineering tool, but struggles with Golang binaries due to symbol stripping. goresym is designed to recover symbol names by parsing Go’s internal metadata, even in stripped binaries. By running goresym on a binary and feeding the results into Ghidra, analysts can automatically rename functions, significantly improving the binary's readability. This approach is especially useful in malware research, vulnerability analysis, and security auditing of Go applications. No debugging symbols are needed. Although manual steps are required, automation via scripts is possible and effective. The integration fills a gap for Go binary analysis in open-source tooling. This pairing increases accuracy and saves hours of work per binary. It’s a must-have combo for any RE dealing with modern compiled Go code.

**⏱️ Time to Use / Best Case Scenarios:**

* When analyzing **stripped Go binaries**
* During **malware reverse engineering**
* When auditing closed-source Go programs
* For **incident response** on compromised Go-based binaries

**⌛ When to Use During Investigation:**

* After importing the binary into Ghidra
* Before or during renaming and code annotation phases
* Once you’ve identified it as a Go binary (via strings, .gopclntab, etc.)

**👨‍💻 Best Person to Use That Tool and What Skills Required:**

* **Target User**: Reverse engineers, malware analysts, vulnerability researchers
* **Skills Required**:
  + Familiarity with Go binary structure
  + Comfortable with Ghidra scripting (Python/Java)
  + Understanding of static analysis concepts
  + CLI usage and light scripting knowledge

**⚠️ Flaws / Suggestions to Improve in That Tool:**

* No **official Ghidra plugin**—manual scripting required
* Limited support for the newest Go versions (Go 1.20+ sometimes not fully parsed)
* No GUI for mapping/importing results into Ghidra
* Suggestion: Create a community-maintained **plugin** for direct integration and update goresym for newer Go runtime structures

**✅ Good About Tools:**

* **Free, open-source, and widely supported**
* Works even on **stripped** binaries
* Makes Ghidra far more useful for Go reverse engineering
* Lightweight and scriptable
* Saves significant **manual effort** for naming and context-building