CS 131 Homework 6
The V Language for Camera-Based HVAC Control

Arnold Pfahnl
University of California, Los Angeles

Abstract

The V language is a promising candidate for the secure, embedded programming requirements of Haversack Inc.'s HVAC control system, SecureHEAT. The language provides an easy way to write clean and easy-to-audit code, the ability to write freestanding programs, has very simple features, and allows for easy interfacing with low-level hardware. However, its use of garbage collection and the language's lack of maturity pose as barriers for the languages's use for the project.

1 Introduction

The Haversack Inc. HVAC control system, SecureHEAT, is a cost-effective solution that utilizes facial temperature recognition to control a building's air temperature. Cameras connect via a wireless network to base stations at secure locations within a building, which do the majority of the computations required for the system due to the lack of processing power in a camera's embedded CPU.

To market the HEAT system, customers with a concern for security including banks, the military, and other organizations need to be sure that the software running on the cameras is as impervious to penetration and information leaks as possible. The following are conditions that must be met:

1. The codebase needs to be cleanly written and easy to audit.
2. The software is a freestanding program that doesn't rely on a separate operating system.
3. The software is very simple without complex features such as garbage collectors or interpreters.
4. The software needs to interface with low-level hardware including cameras and network interfaces.

This report will delve into whether the V language satisfies these requirements well enough to the extent that it can be recommended as the programming language for implementing SecureHEAT.

2 The V Programming Language

The V language is summarized succinctly by its official tagline,

> *Simple, fast, safe, compiled. For developing maintainable software.* [5]

The language is statically typed, compiled, and its syntax is similar to Go, a language inspired by and strives to improve upon the features of C.

2.1 Security

V is a language built on simplicity, and this translates directly into a very safe language. V doesn't have nulls, global variables, undefined values, undefined behavior, or variable shadowing. It utilizes bounds checking, makes variables and structs immutable by default, requires pure functions by default, and has mandatory error checking. The language is designed in a way that makes programming errors difficult to make.

Embedded developers often use global variables to minimize memory usage and for speed increases. However, most use cases for global variables aren't justifiable, especially since they compound programming errors and security issues, and should be limited to static physical properties (e.g. the number of cylinders in an engine being controlled) or system state information that should only every be declared in one part of the codebase [2]. For the few cases that globals are necessary, the `-enable-globals` compiler flag enables the use of global variables.

V's clean code paradigm makes auditing a V codebase simple and safer. Oftentimes, there is only one way to do something, and aspects of the language such as throwing a compilation error when a program contains unused variables ensures that there isn't much extraneous code. `vfmt` is a built-in tool that formats code in a standardized way, so implementing style guides isn't necessary. This, combined with the built-in profiler, testing functionality, and automatic

documentation forms the tooling that makes V consistent to audit and easy to write clean, and safe code in.

By promoting the use of simple value types, string buffers, and simple abstraction-free coding styles instead of the use of objects requiring memory allocation, V tends to be less prone to memory security flaws such as buffer overflows. The burden of freeing objects is handed off to the V compiler, and this also decreases the risk of a programmer introducing security risks with memory leaks.

2.2 Usability and Reliability

The V language is fairly easy to use. From the documentation, the language's developers claim that nearly the entirety of V can be learned within an hour [4]. V is similar to Go, so any developer with experience in Go will have no trouble picking up V. Go and V are syntactically similar to C, so Haversack's transition from using C and C++ to V would be relatively easy. Although the documentation is very simple, it also leaves a lot to want, and some consider it lacking maturity [1]. This coincides with the growing pain of a new language, and this can also be seen in the lack of mature libraries.

Flexibility and generality is also a strong suit of the language. The documentation specifically mentions V being powerful for software development in areas relevant to the SecureHEAT project such as embedded, systems, drivers, and mobile. A notable feature is the ability to cross compile by compiling code that is intended for use on another platform on nearly any other platform. For SecureHeat's embedded freestanding programming objective, this is an extremely useful feature.

The V language provides easy mechanisms for interacting with hardware at a low level. One of these features is the language begin heavily interoperable with C. V was designed as an iteration on C, and C code can be directly included in V. V also allows for inline assembly that allows for interfacing very close to the hardware level. The compiler also provides the `freestanding` option that allows for freestanding programs to be generated, a requirement of the SecureHeat project [4].

2.3 Performance

Memory management is fairly fast with V. As mentioned in the subsection on security, V discourages unnecessary allocations, and most objects that are allocated are freed by the autofree engine, a system where the compiler inserts free calls automatically [4]. There is, however, also a small percentage of objects that can't have inferred free calls, so reference counting is used for these remaining objects. From the documentation, the extent to which reference counting can have an impact on performance isn't stated.

Programs written in V are just as fast as C since V's main backend compiles to human readable C. There's no performance hit for calling C code in V, and the language lacks runtime reflection with its built-in serialization that typically makes programming languages like Java slower.

The V language also compiles to native binaries without any dependencies, and this typically makes V's binaries very small. An example statistic provided by the V langauge team states that "a simple web server is [implementable with] only 65 KB" [5].

3 Summary

Overall, given the conditions that need to be met for the SecureHEAT project, the V language might not be a good choice. One sticking point is the lack of maturity of the language, and the claims that are made with little in terms of actual usage statistics to back the claims up. There are also inconsistencies on the information available about the language such as the languages's developers saying that there is no garbage collector on their language comparison webpage even though the documentation clearly states that reference counting is used for memory management [3, 5].

More importantly, the language does not satisfy the specific condition that garbage collectors cannot be a part of the software. The V language utilizes reference counting, and this is a form of garbage collection, so this automatically rules out the V language from use. However, if all other programming languages under consideration don't satisfy all constraints, the V language is a promising candidate since the language provides a means to write clean and easy-to-audit code, the ability to write freestanding programs, has very simple features (ignoring garbage collection), and allows for easy interfacing with low-level hardware.

References

[1] Ben James. The v programming language: Vain or virtuous. https://hackaday.com/2019/07/23/the-v-programming-language-vain-or-virtuous/. Accessed: 2021-06-01.

[2] Phil Koopman. Minimize use of global variables, Better Embedded System SW. https://betterembsw.blogspot.com/2014/06/minimize-use-of-global-variables.html. Accessed: 2021-06-01.

[3] V. Comparison of v and other languages. https://vlang.io/compare. Accessed: 2021-06-01.

[4] V. V documentation. https://github.com/vlang/v/blob/master/doc/docs.md. Accessed: 2021-06-01.

[5] V. The v programming language. https://vlang.io/. Accessed: 2021-06-01.