Evaluating a language for secure camera-based HVAC control

Ethan Wong
CS131
Discussion 1C

Abstract

In this assignment, we are tasked with researching the fairly modern V programming language (Vlang) as a potential foundation for a new state-of-the-art HVAC system known as secureHEAT. This heating, ventilation, and air conditioning system will be used in place of traditional thermostats and will operate by determining the temperatures of people in the building to determine what the ambient temperature should be set at. This report will analyze the strengths and weaknesses of Vlang, as well as exploring language-relevant technical challenges that will come up when implementing the application that controls secureHEAT.

1. Introduction

Vlang was released in 2019 as an alternative to the popular programming language "Go". Designed to be a multi-paradigm programming language, Vlang is statically typed and has its own garbage collector. The main reason Vlang was created was because the developers wanted to combine the excellent performance of C/C++ with the safety features seen in more modern programming languages. Vlang itself is quite a simple language, with easy-to-understand syntax and minimal abstractions. Vlang is also well-known for being incredibly fast and efficient in a variety of different situations.

2. Analysis of Vlang

This section will analyze some of the most important characteristics for any programming language. Topics include security, ease of use, flexibility, performance, and other general traits about Vlang and how they will benefit or harm the implementation of the secureHEAT application.

2.1 Security

Security is a very important issue for secureHEAT, especially in high-security buildings such as banks, military facilities, or prisons. It would be best if Vlang were a completely freestanding programming language or could at least create programs that are freestanding. Unfortunately, security is one of Vlang's weaker points. While there are no glaring issues with the security, there are potential issues with the garbage collection system. The garbage collection system is based off of something called the "autofree engine", which is basically just an automatic garbage collector. Features like these are quite high-level and may garner disapproval from clients since high-level features tend to have more gaps that malicious intruders can exploit. While autofree can be disabled manually by the developer, it is important to note that the very existence of the feature within the language can lead to potential exploits. In addition, V is generally expected to work with an operating system. It is possible to create V programs without an operating system, but many of the features rely on one. For example, one of the best characteristics of V is that it can be cross-compiled across different operating systems. This is definitely beneficial in helping teams of developers work together but it can also prove to be a security risk. V is decently secure from a typical programmer's perspective but some of its high-level features may be a liability for high-security purposes.

2.2 Ease of Use

While Vlang may definitely have some security flaws, it has many other commendable traits. Among them is how easy the language is to pick up and learn. One of Vlang's selling points is how similar it is to the Go programming language. The official website even claims that "If you know Go, you already know ≈80% of V". This is great for developers as they will likely have at least a basic understanding of Go, allowing them to pick up Vlang quite quickly. The syntax is also reasonably simple. There are no extremely odd features (such as all the parentheses needed in Racket, for example) that will make code difficult to read or particularly difficult to understand. This is also a plus as it allows developers to spend more time creating the application instead of worrying over syntax issues that will waste their time. Finally, there will not be any fuss about code formatting like there is in many other languages such as C/C++. There is a single

official coding style that the vfmt formatter will enforce, making code clear and concise for all developers.

2.2 Flexibility

Perhaps one of Vlang's strongest suits is its incredible flexibility. V, like many other languages, was heavily inspired by C/C++. In fact, Vlang's backend actually compiles to human-readable C. It will not be difficult to look at low-level code (such as code for hardware like cameras) as it will be portrayed in a simple fashion. Because of this, V can also call C code if needed. This is favorable as C has more libraries and other niche features that may not be included in V yet. The fact that V is so closely intertwined with C is valuable for developers since they will most likely have a solid understanding of C.

A very nice additional feature is that V programs can be cross-compiled with Windows and Linux machines. Unfortunately, there is cross-compilation for macOS yet, but having Windows and Linux available means that a large majority of developers will be able to work together on their own machines. This is advantageous as developers can program and collaborate no matter which operating system they prefer. In addition to that useful compilation feature, Vlang has a tool called "hot code reloading". This means that changes made to Vlang programs can be seen without recompiling all the code. This may not seem like it makes any difference, but it can save precious time when dealing with enormous programs

V is a relatively new programming language so there are still many important flexibility features that are still in development. As mentioned before, there is currently no cross-compilation for macOS, but that will be released soon. Also, there will be support for iOS and Android development soon. This will be a neat feature that could help this application in case mobile devices are ever needed to control the secureHEAT system. Finally, one of the most interesting features that has yet to come out is the C++ to V translation. Once released, this tool will allow developers to transform their previously created C++ programs into V code. This is huge in case developers are too lazy to learn V, as they can just program in C and translate it once they finish. This will grant them the ease of writing C code that they are familiar with while obtaining the flexibility and performance of Vlang.

2.3 Performance

Another one of Vlang's strongest points is its amazing performance. Because Vlang's backend compiles to C, that means that it shares the same performance as C without all of inconveniences that come along with C programming. A large part of Vlang's efficiency is that it cuts out a lot of overhead. For instance, it has minimal allocations which saves a lot of space in memory and also eliminates the time wasted allocating objects in the first place. There is type inference (like in Python) that saves the user time so that they do not have to manually declare the types of their variables. This harms performance a minutely, but not to the point that V is as slow as Python.

Compilation has incredible performance for V. The V compiler itself can be compiled in under a second. This is remarkable when compared to some other compilers like GCC, which takes approximately 50 minutes to compile, and Clang, which takes roughly 60 minutes to compile. The V compiler is also very small in terms of storage as it is less than 10MB. For comparison, GCC is 8GB and Clang is 90 GB. Vlang is much more compact and therefore much quicker. Also, V programs themselves compile at an incredible rate. If using a Clang backend, V can compile about 110,000 lines of code per second per CPU core. With an x64 backend, this rate increases almost tenfold to approximately 1,000,000 lines of code compiled per second per CPU core.

2.4 Miscellaneous

There were a few other features of Vlang worth mentioning that did not quite fit into any of the other categories. First, there is a built-in testing framework and a built-in code profiler that help developers easily analyze and debug their code according to test cases. There are also libraries that can help deal with camera controls, something that would be incredibly useful for the secureHEAT application since it heavily relies on cameras to monitor the temperatures of people in a building.

3. Conclusion

Overall, V is a decent choice for the implementation of the secureHEAT application, but it is not perfect. The number of good features heavily outweighs the number of bad features. However, because there are possible security exploits from the memory management system, customers may be hesitant to accept secureHEAT in high-security buildings. These security concerns must be addressed either by the having the developers implement sufficient security measures, or by using a different language that can create truly freestanding programs that are more secure.

References

V Documentation
https://vlang.io/
https://github.com/vlang/v/blob/master/doc/docs.md
https://modules.vlang.io/
https://vpm.vlang.io/

General Information
https://web.cs.ucla.edu/classes/spring21/cs131/hw/hw6.html
Professor Eggert's Lecture Notes
Discussion 1B Slides