CS131 HW6 Summary on V

Brendan Rossmango
Discussion 1C

1. Introduction

In this paper, we investigate the feasibility of using the V programming language for Haversack Inc's SecureHEAT application, which is a system that innovates controlling temperature more efficiently in buildings. In addition, SecureHEAT has the potential of saving a significant amount of money. SecureHEAT's crafty approach uses affordable cameras to implement a system, called Human Embodied Autonomous Thermostat (HEAT), that monitors the faces of humans in the building and calculates their temperatures to control the building's overall temperature.

The cameras are connected to base stations via a wireless network and have no battery backup. Since the cameras' embedded CPUs have limited processing power and memory, the base stations do most of the calculations to control the building's heating, ventilation, and AC units.

The major concern with SecureHEAT's system deals with the security issues of the cameras; specifically, there is potential that the software in the cameras is penetrated, and many possible customers like banks and intelligence services require high security. Additionally, Haversack has used a combination of C and C++ to program such software in the past, and these programs are vulnerable.

The software used for SecureHEAT must erase any vulnerabilities. Furthermore, the software must be cleanly written and easily auditable, as simple and stripped-down as possible (which rules out higher-level languages), and capable of interfacing to low-level hardware devices like cameras and network interfaces. Additionally, it should be a free-standing program with no separate operating system.

We evaluate V as a candidate for this software by looking at its features regarding security, ease of use, flexibility, generality, reliability, and performance.

2. Overview of V

V is a language that prides itself on being very small, very fast (as fast as C), and very simple; its official website says one can learn the entire language by reading the documentation in under an hour. It is a multi-paradigm language that intends to combine a high degree of safety with the performance of C/C++, and it uses a syntax very similar to Go's.

3. Security

Since the SecureHEAT system uses people's facial temperature to calculate the building's temperature, the software behind SecureHEAT's cameras needs to be invulnerable to attacks to protect the customers' confidentiality.

One of V's hallmarks is its safety and security; V has no null, global variables or state, undefined values or behavior, or variable shadowing (cannot name a variable the same name of a variable that is already in the parent scope), unlike C, which is vulnerable for many of the above reasons. Furthermore, by default, arrays are bound-checked, variables and structs are immutable, and functions are pure. To make a variable mutable, one uses the keyword 'mut'. Bound checks prevent buffer overflows which is a vulnerability in C and making variables and functions immutable and pure respectively prevents unwanted side effects that could lead to potentially insecure code.

V is also strongly typed, unlike C, with one exception being automatic promotion for certain numeric values.

Instead of using throw/try/catch blocks, V handles errors using Option types, and there are four ways to handle Options and return errors. Ease in handling errors is important for security concerns.

V supports writing memory unsafe code but not by default. More importantly, it requires that any potentially memory-unsafe operations be marked intentionally with the keyword 'unsafe'; examples of operations that require being marked by 'unsafe' are pointer arithmetic and indexing, conversion to pointer from an incompatible type, and calling certain unsafe C functions. However, V's documentation notes that unsafe blocks are "a work in progress," and they do nothing to stop users from writing unsafe code; the code is simply marked 'unsafe'. Since handling memory-unsafe operations is still a "work in progress", an attacker can learn what memory-unsafe operation V does not still handle correctly; this raises concerns.

For the most part, V is a very safe language that succeeds in eliminating most vulnerabilities that unsafe languages like C/C++ have (global variables and state, undefined behavior and values, nulls, etc.). However, the biggest problem with V's security is its interoperability with C code; V can call C code and include libraries from C, and some of these libraries use APIs that are potentially dangerous. V requires that potentially memory-unsafe operations be enclosed in an unsafe block, but attackers can still use V to call unsafe C code and its insecure APIs, like free, strlen, and strncmp. This alone is a very big red flag for using V in software that is supposed to be very secure.

4. Ease of Use and Generality

V is a very simple language, and it is very quick to learn V from its documentation. V has minimal abstractions, clear code, a small feature set, and is generally very easy to understand.

V includes many features that make it easy to use and have a high degree of generality; it is very programmer-friendly like Python. V has string interpolation and string concatenation akin to Python and makes array manipulation easy: one can append to the end of an array using the push operator <<, check if a value is in an array using the in operator, copy data using clone(), check if the array meets a condition using any() or all(), and filter and map an array using the eponymous methods. There is also array slicing.

V also has pattern matching (in the form of using 'match' to match sum types that can hold a value of several different types) and anonymous and high order functions.

V's generality is best seen in having only one keyword for looping: 'for'. For loops can be like in Python (for/in), like a while loop with a single condition (i.e., for i < 100), with no condition (which is an infinite loop), or like in C (for assignment; condition; update). This lends to generality (since there is no need to have other keywords for loops that achieve the same thing) and flexibility since there are multiple ways to use a for loop.

5. Flexibility and Reliability

For flexibility, V enforces one style to write code in with the 'v fmt' formatter; this makes code easier to read and maintain at the cost of allowing multiple different styles.

One pro of V is that it can call C code and can be called in any language that has C interoperability. Additionally, C code can be translated into V.

V avoids doing unnecessary memory allocations by using value types and string buffers and promoting a simple abstraction-free code style. For example, using the 'cap' feature in an array (which reserves memory for the number of elements specified by the capacity 'cap'), reallocations can be avoided when appending to the array, and performance is improved. This is useful due to there being a very limited amount of memory available for the camera software. Arrays are more flexible in general due to the optional 'init' parameter when making an array to make the array have a default value specified by 'init', in the same manner as Python's default dictionary.

Like Python, V's functions can return multiple values, which adds flexibility since one has the choice to use all the return values or discard some of them.

V has some reliability issues, especially regarding memory management. V is not completely stripped down since it has a garbage collector; therefore, it is not as low-level as can be and takes up a lot of memory that alternative programming languages would not. Most objects are freed by V's autofree engine, where the compiler inserts free calls automatically during compilation; the remaining objects are freed via reference counting. However, one can disable the use of the garbage collector with the -manualfree option when compiling, so the programmer can strip V down a bit more and not waste memory for the garbage collector.

6. Performance

V has fast compilation and performance. It compiles over 1 million lines of code per second per CPU core. It takes less than 10 MB and 1 second to build its compiler. As for its performance, it is as fast as C, as its main backend compiles to human-readable C. It has cheap interoperability with C and a minimal number of allocations. Furthermore, V can emit human-readable C, so it can be optimized using GCC and Clang. V also has a built-in code profiler to get a detailed list of all function calls, including the average time and total time per call. Finally, it compiles to native binaries without any dependencies; it is well suited for low-level purposes just as C is because it does not produce large binaries, nor does it use a lot of memory.

One issue with V is that it uses reference counting for objects that are not automatically freed; the documentation says ~90-100% of objects are freed automatically. Reference counting is a slow method for garbage collection because reference counts of each object must be incremented or decremented for each reference or dereference, and each time the reference count must be checked if it is zero. However, reference counting does not use much memory as a garbage collection approach.

V makes use of concurrency and threads, but with no OS to schedule them, this is of no use for the software.

7. Conclusion

V is simple, fast, and comes with minimal abstractions to produce a sleek programming language. It is cleanly written with an enforced style, easily auditable, capable of interfacing with low-level hardware due to its low-level nature and interoperability with C (so it can include C's libraries to interact with cameras and network devices), and for the most part is stripped down, with one exception being that is has a garbage collector that it uses to free some objects. However, one can disable this automatic memory management by compiling with the -manualfree option. Additionally, V is easy to use with various features like pattern matching, generics, and more. Unfortunately, its one fatal flaw in its potential to be used as the language for the software of SecureHEAT's cameras is its security, specifically its ability to include C code and library APIs that are not memory-safe. Attackers can penetrate the cameras' software by using V to call vulnerable C code, a vulnerability that Haversack Inc. already eliminated by choosing to not use its combination of C/C++ code. For this reason alone, it is a good idea to research alternative programming languages as candidates to be used for the software of Haversack Inc.'s SecureHEAT's cameras, since V may not be the best choice.

8. References

[1] University of Michigan. *Turning faces into thermostats: Autonomous HVAC system could provide more comfort with less energy*. 2020.

https://www.sciencedaily.com/releases/2020/06/200616083353.htm

[2] Da Li, Carol C. Menassa, Vineet R. Kamat, Eunshin Byon. *HEAT - Human Embodied Autonomous Thermostat*. 2020 https://ebyon.engin.umich.edu/wp-content/uploads/sites/162/2020/05/2020.HEAT-Human-Embodied-Autonomous-Thermostat.pdf

[3] *The V Programming Language*. https://vlang.io/

[4] *V documentation*.

https://github.com/vlang/v/blob/master/doc/docs.md

[5] Ben James. *The V Programming Language: Vain or Virtuous?*. 2019.

https://hackaday.com/2019/07/23/the-v-programming-language-vain-or-virtuous/