CS131 Homework 6: Evaluating the Language Odin for a HEAT System

Belle Lerdworatawee
Discussion 1C

Abstract

Haversack Inc. is developing a new system to control building temperatures more efficiently. This paper evaluates the effectiveness of the programming language Odin as a means to accomplish this. It will discuss concerns with respect to security, ease of use, and performance.

1 Introduction

Haversack Inc specializes in manufacturing HVAC systems for customers. They're working on a project called SecureHEAT that uses inexpensive cameras to implement a Human Embodied Autonomous Thermostat (HEAT) system. The system monitors occupant faces, calculates their facial temperatures, and adjusts the air temperature of the building accordingly. Due to the camera's limited hardware, they can merely capture data. They connect to base stations through a wireless network; these stations perform the calculations and control the temperature.

Previously, the software inside the camera was written in C and C++. Potential clients are concerned about security attacks and information leakage. In order to address these concerns, the software must be readable, freestanding, and capable of interacting with low-level hardware devices. Odin is a potential replacement for writing the software.

2 Odin

This language was created as an alternative to C with the goals of simplicity, high performance, and the joy of programming [1].

2.1 Features

The creator Ginger Bill asserted that there "ought to be one way to write something" [2]. So semantically, Odin is less expressive as compared to other languages. For the purposes of the cameras, this isn't an issue. The program doesn't require much creativity as it doesn't handle temperature calculations. In fact, standardized code would be easier for other developers to improve it and this limits undefined behavior.

Some practical features are package systems, the keyword foreign, implicit context systems, and parametric polymorphism. Every Odin program consists of packages that make importing code easy and suits hardware with limited memory [3]. In this manner, we can import only necessary modules, and reduce strain on the cameras' limited memory. While Odin doesn't have much native support for low-level systems, it has a keyword `foreign` to import libraries written in other languages [3]. This is especially useful as C has many libraries for working with low-level devices. Programmers reap the benefits of Odin's high performance while being able to utilize established C functionalities. Next, Odin has an implicit context system, which is a builtin implicit variable named `context`. This allows developers to modify third-party and library code [3]. `context` could be useful for changing how a library allocates or logs data for our specific purpose of monitoring occupant faces and transmitting that data. From a programming perspective, this is simpler than C which achieves the same task by overriding library defining macros. There's a risk though that incorrect changes are made which could introduce bugs into the software. Lastly, Odin supports parametric polymorphism, which adds flexibility. This feature eliminates code duplication, which again is necessary due to the cameras' limited memory [3].

2.2 Security

Although Odin has many promising features, it has several security limitations. It has a builtin type called `any`, which can reference any data type. Internally, `any` is implemented by creating a pointer to the underlying data [3]. It was created to make printing procedures more convenient. The issue with this is that the compiler can't assume anything about the `any` type and thus, developers have to worry about type safety. While `any` offers flexibility, it's not necessary for our application and could actually make it easier for outsiders to tamper with data.

In addition, Odin users must manage their own memory, allocations, and tracking. An upside to this is that Odin offers much support for custom allocators through the implicit context system [3]. Good memory management is essential for the cameras to have fast response time and scan faces quickly enough. By manually managing memory, we gain fine-grained control over memory and can more directly improve performance. However, the software is also more error prone. In particular, this draws large security concerns as we risk losing data or accessing illegal areas. In this vein, Odin shares the same security risks as C and C++ in that it can also be exploited to execute malicious code or leak confidential information to adversaries.

On another note, Odin lacks exceptions which actually increases reliability. Another useful feature is that Odin procedures can return multiple results; they can return an extra variable of error type. This value can be tested explicitly from the call by means of a switch statement [4]. In this way, we know exactly what and where the error stems from while exceptions typically produce vague, unhelpful messages. This is advantageous for SecureHEAT as Odin programs are easier to audit and debug.

2.3 Performance

One hallmark of Odin is its high performance, and several factors contribute to this. For one, developers can improve performance by disabling array bound checks with the `#no_bounds_check` directive [3]. However, this isn't recommended as there's no guarantee that the camera software won't accidentally access out-of bound data. Additionally, hackers gain an opportunity to insert and execute malicious code in these out-of-bound areas, and security is a priority.

In addition, Odin lacks a garbage collector which has both performance and security benefits. Clients don't need to worry about information leakage and the camera won't waste its limited processing power on clearing memory. Returning briefly to Odin's lack of exceptions, this boosts performance as exceptions require unwinding the stack which is much slower than Odin's method of testing the return value.

Lastly, Ginger Bill shared a software called EmberGen by JangaFX that's a real-time volumetric fluid simulation tool [5, 6]. Users can instantly simulate and render graphics like smoke [6]. Since a powerful rendering tool was built with Odin, this implies that it works well enough with GPUs to be able to produce well-rendered simulations in real-time. We can trust then that Odin is suitable for working with hardware.

3 Strengths and Weaknesses of Odin

The main strengths of Odin for our proposed application is that the language provides high functionality despite being stripped-down. Odin programs don't require an operating system and have no garbage collection so clients can trust that the program is less vulnerable to attacks. Additionally, it has familiar, convenient data features like splicing and dynamic arrays. Overall, the syntax is relatively easy to learn and allows us to take advantage of libraries in other languages.

On the other hand, Odin's most notable disadvantage is the lack of community support. It was difficult finding other security applications written in Odin to compare. Additionally, having to manually manage memory places large responsibility on the programmas at the expense of performance and serves as a security vulnerability.

4 Conclusion

When choosing a language for an application, many aspects need to be considered. After careful deliberation, I would not recommend Odin as a replacement for SecureHEAT's software. While it did fit many of the requirements such as allowing freestanding programs, readability, and ease of use, its security issues outweigh these benefits and it has to rely on outside libraries for hardware support which is inconvenient. However, Odin holds a lot of potential and with further development, could become a stronger candidate.

5 References

[1] Odin. The Odin Programming Language. https://odin-lang.org/. Accessed: June 4, 2021.

[2] Ginger Bill. gingerBill. Odin Programming Language: An Introduction - 2020-11-26. https://www.youtube.com/watch?v=rCqFdYUnC24, 2020. Accessed: June 4, 2021.

[3] Odin. Odin Overview. https://odin-lang.org/docs/overview/. Accessed: June 4, 2021.

[4] Ginger Bill. Exceptions — And Why Odin Will Never Have Them. https://www.gingerbill.org/article/2018/09/05/exceptions-and-why-odin-will-never-have-them/, 2018. Accessed: June 4, 2021.

[5] Ginger Bill. gingerBill. Talk & Demo - A New Programming Language. https://www.youtube.com/watch?v=TMCkT-uASaE&t=6252s, 2016. Accessed: June 4, 2021.

[6] JangaFX. EmberGen. What is Embergen?. https://jangafx.com/software/embergen/. Accessed: June 4, 2021.