VeriServ Proposal
by Oreoluwa Babarinsa, Alex Liu, Pulak Goyal and Zach Fogelson

General Vision
The goal of this project is to produce a minimal, verified web-server. The inspiration for this project is that web-servers share many challenges with operating systems with regard to potentially malicious user-input, high performance necessities, and persistent security threats. As a result, I think that web-servers could benefit from a micro-kernel architecture as well as from being formally verified.

Maximal Goals
If this project succeeds wildly, we would hope to produce the following. First, we would want to have a specification and corresponding semantic invariants for web server such that if one were to design a web server according to our specification which met our invariants, said web server would be durable to all but the most arcane forms of attack. Further, we would like to have a proof-of-concept implementation of said specification in C, which has been fully verified by Frama-C to show that we have no crashing or security bugs, nor any unexpected behavior due to broken invariants. Interesting invariants we seek to prove include request isolation (i.e. receiving two requests concurrent shouldn't produce incorrect behavior).

Minimal Goals
The absolute minimal goal of this project is to verify a few interesting properties of a simplistic web-server such as having making sure that it obeys security configurations (i.e. if a route is off limits, than it is provable that no one can access that page).

Concerns
There are two large concerns for this project. Firstly, is the very difficulty of implementing certain parts of a web server. Parsing of any sort, especially HTTP Parsing, is a difficult and exacting tasks. This is doubly so when one wants to do it in C, and thus introduce the possibility of buffer overruns. We’ll likely spend a fair amount of time attacking this one section of code.

The second main concern is sockets. Sockets have numerous failure modes, many of which are not obvious and can be utterly silent. Making code that is robust up to any and all network failure is going to be, for lack of a better term, a pain.

Tools
The central tool for this project will Frama-C. Time allowing, we may use KLEE to attack particularly sensitive sections of code, such as our parser, but that will come after we are already fairly confident in our implementation and have already proven key invariants about the code.

Related Works
Verified Security for Browser Extensions - A. Guha
Verified Systems by Composition from Verified Components - F. Xie

One Week Project
- Implement a very minimal web server
- Prove that this web-server is free of crashing bugs and memory errors
- Prove that this web-server has abides by a given security policy.

Collaboration Plan
We generally plan to have two people focused on designing the server (potentially even implementing a model system in a higher-level language). These two people will also develop the system invariants.

The other two people will focus on implementing the web server, as well as actually trying to prove the invariants given by the first team. If the invariants can’t be proven or they can develop malicious test cases which abide by the invariants given by the first team but produce bad
output/crash, then the first team has to rework those invariants.

One could conceive of this in quasi-mathematical terms in that the first team is proposing a theorem of how to produce a rock-solid webserver, and the second team is a something of an adversary trying to implement said the specification of the web-server in such a way we get security vulnerabilities or other undesirable effects.

We will likely shift people between teams throughout the project as the balance of work shifts between designing and implementing.