Contact

Rob Fielding
Senior Developer
rob.fielding@gmail.com
http://github.com/rfielding - Public code repository (nothing work related)
http://rfieldin.appspot.com - Demonstration of my work with Wizdom - high performance mobile development

Summary

I am currently working at firewall vendor Check Point Software, working in the context of Intrusion Prevention and general Security. My experience spans from J2EE and web development and thick client C# Windows, to low level C in Linux servers. I have been using Java since HotJava Alpha came out, and have undergone Sun Java2 certification.

Education

George Mason University - Computer Science, graduated 1996
Focus on algorithms, 3d graphics, assembly language, C
One of the earliest Java users (HotJava)

Skills

Used professionally on a daily basis: C/C++, Java, C#, Python, Javascript, PHP, Objective-C, graphviz, VMware (some use of the APIs), git, DHTML/CSS/JavaScript, Wireshark/tcpdump, C#/WPF

Also used: Go, Erlang, OCaml/Haskell, Spring MVC, ExtJS, Cassandra/Thrift, Solr, Lua, Hadoop MapReduce, Django, Spring/Hibernate, Arduino.

Employment

2007-Present: Sofware Developer - Check Point Software Technologies

Check Point acquired NFR Security (roughly 20 employees)
Currently working on a (in kernel module) virtual machine language for use within the firewall, and IP packet parsing in a specialized hardware accelerator. Added simulation and unit testing support into the hardware acceleration code (C IP packet parsing) that was deeply tied into running only on real hardware.

Managed Security Services Portal, which started out as a PHP/Postgres site built by sales engineers and handed to us to scale up before new customer load created a performance problem; a classic high throughput streaming application (incoming firewall logs from all over the world). This used a Postgres database a 2Tb working set (not including the replica), where we had to solve a lot of big performance problems while not violating the SLAs. It was migrated to use a J2EE server (using Spring MVC to implement REST services consumed from PHP that emits the necessary JSON/XML/HTML to go with the static CSS/ExtJS/JavaScript) backend, and much of the database schema reworked to be much more efficient - on the live database. Much of this involved re-working database indices, migrating data to more efficient representations, eliminating inefficient summary database queries in favor of heirarchial/queryable counters in the Java server, and reworking view rendering to render data in bounded memory space.

Worked on a C#/WPF based IDE and debugger for our custom language (cpcode/ncode) which is used in a Linux kernel module context

Sensornet work was a classical log processing application involving ExtJS and cgi, writing MapReduce jobs to generate IP connectivity graphs

My group is the group that introduced git usage into the company, so we are familiar with the difficult parts of using it on a daily basis (dealing with the consequences of choosing merge in cases where there should have been a rebase, interactive rebasing to create clean histories, etc),we were using Gerrit as our workflow for about a year as of last year.

IPS-1 Security Alerting Dashboard (continuation of Sentivist Server)
We had done research into a J2EE backend plus C#/WPF infrastructure to handle automatically exposing async client APIs to sync remote EJBs, using Hibernate/Spring as the backend, and writing a C# implementation of JavaSerialization to support EJB's RMI usage.

2011: Independent Developer - Wizdom Music (Part Time)

iPad/iPhone development of a real-time music instrument for professional use.
The app names were: Mugician, Geo Synthesizer, and Cantor.
Geo Synthesizer was not only useful and popular, but financially successful for our team of three people (A well known rock musician with his programmer, and myself).
This work involved solving a lot of performance problems in C code. Battery life, memory consumption, SIMD parallelism, signal processing, and hard real-time constraints dominated every aspect of this design and development.
This experience involved a lot of running the business directly, and involving customers directly in design and maintenance.
The source code for all of this, except for Geo Synthesizer (which may now be owned by Roli Labs), is in github.

2004-2007: Developer - NFR Security / Network Flight Recorder

Sentivist Server - a high throughput (MySQL tens of millions of alerts per day 40Gb working set, per sensor device) JBoss/J2EE backed application server, deeply involved in every aspect of its implementation and maintenance
Sentivist Dashboard - a Java Swing client that connected to Sentivist Server, deeply involved in every aspect of it. This was a shrink-wrapped CD installed product that had to be bug-free enough to withstand an update cycle as low as 1 year

We got a patent related to viewing streaming data: 11/586,689, one of many useful innovations in the security alerting dashboard

Because we were dealing primarily with high throughput streaming logs, our J2EE app was largely based on JMS and merging it with queries to give us an API similar to what is seen in Esper today. We also had to make some rather drastic changes to Jboss4.x so that we could have the client connect correctly to the server across NATs, tunnels, and port forwards; while completely eliminating cleartext connections between client and server. For performance reasons, this involved changing things to use immutable data structures that resemble modern CRDTs, and using the largest batches that latency constraints would allow. We had to rewrite the transaction system to avoid deadlock states in the MySQL server and in objects running in the JVM (ie: calls to remote EJBs needed all remote calls to declare every lock that could be taken so that they all get taken in a globally consistent ordering). This meant that our "J2EE" application was rather unorthodox, not using J2EE persistence, using POJO to database APIs that correctly handled asynchronous design and efficient serialization; making our app more resemble an Erlang application than a typical J2EE application.

1999-2004: Developer - Logicon / Northrop Grumman / ComTek / DISA

Worked on multi-machine application suites primarily providing APIs wrapping up LDAP for DoD PKI applications. This involved client-side SSL authentication (software and smartcard tokens). At one point, installations got reduced from 30 days due to waiting periods for certificate signers happening at random installation points, down to a few days by building a tool to request all future required certificates in one step and write them to the formats that the applications will demand later on
Worked on SAML authentication based applications related to DoD PKI
Taught class on encryption and PKI (similar notation to Bruce Schneier's Applied Cryptography group, but more group theory oriented notation), authored a lot of the documentation related to demonstrating that we took required measures to secure our code

Authored a Java Servlet based UI toolkit (similar to Echo2 framework before it became JavaScript based) to deal with scenarios where JavaScript was not yet allowed for security reasons, yet have a rich widget API which was used in our applications.
DoD Secret clearance

1997-1999: Developer - Federal Reserve Board Of Governors

Worked on banking surveillance applications. Generally, work was done in Powerbuilder (and occasional bits of Win32 C API). While I was there, I introduced the use of Java for these kinds of applications, and the first use of Java in production was by way of Silverlight.

Other

Willing to travel. (My current work has had quite a bit of international travel. Usually to work on-site with remote co-workers.)

Though I am currently doing rather low level work, my interests have been centered around good high level tools such as strongly typed languages, and using other measures to have machine checkable confidence in code (ie: DBC, LanguageTheoreticSecurity, etc).