Avraham “Abe” Bernstein | Master S/W Engineer | CV-Abbrev

Represented by Wizedom Corporate & Technical Recruiting

Last update: 2017-09-27

0.1 Contact Info & Links

email: Avraham.Bernstein+cv+wizedom@gmail.com [1]

geolocation: Jerusalem Israel, **tz:** UTC +02:00/+03:00 [winter/summer]

tel-IL-mobile/whatsapp: +972.54.641-0955

tel-US-mobile: +1.845.402-0023

www-home: http://purl.org/Avraham.Bernstein

linkedin: https://www.linkedin.com/in/AvrahamBernstein

cv-abbrev: HTML, DOCX, PDF **[this file]**

cv-full: HTML, DOCX, PDF

1.0 Summary

I am an experienced computer scientist and S/W architect. I have devised innovative solutions to many S/W problems for a wide range of fields, including:

- cybersecurity
- cryptography
- bioinformatics
- transportation vehicle route guidance
- factory automation
- automated testing
- blind vision
- accessibility
- telecommunications
- VLSI CPU design

I have worked for a number of organizations, large and small, and helped them realize improvements in their product performance, often putting them in the front rank in their field. I have acquired expert knowledge in a number of fields, often liaising with noted experts, and have been able to quickly apply this knowledge to improve the competitive position of the companies and their products. I have a keen interest in computer languages, both practical and theoretical. I have created a number of

---

1 For security reasons to protect against email harvesting, the email address has been obfuscated.

domain specific languages (DSL) that were instrumental in greatly simplifying seemingly intractable problems.

In order to understand my S/W design principles, see the following appendices (in the full version of my CV):

- Programming Language Preferences and Musings
- Domain Specific Languages
- How To Write Correct, Maintainable, Secure, and Easy-to-Test Code

2.0 Work Experience

2017-present: Cybersecurity Consultant

@Self-Employed, Jerusalem

Keys: cybersecurity, architect, algorithms, obfuscation, compiler, C/C++, javascript, WASM

1. I am developing an obfuscating compiler for C/C++ and for Web Assembly (WASM). Still in stealth mode.
2. I am a security mentor for the Jerusalem Mass Challenge start-up hub.

2011-17: Viaccess-Orca: Security Policy Mngr & Architect: Cybersecurity: OTT Internet Pay TV System

@Viaccess-Orca, Ra'anana - a subsidiary of @Orange France, and @Discretix/Sansa Security, Netanya recently acquired by @ARM

Keys: cybersecurity, DRM, architect, algorithms, anti-reverse engineering, obfuscation, LLVM compiler, cryptography, data science, fuzzy logic, C/C++, TCL, Python, bash, Android root detection, Linux, ELF edit, IOS

1. The product was an Over-The-Top (OTT) Internet pay TV system. We provided the S/W infrastructure to our customers, the legacy (i.e. satellite and cable) pay TV operators, so they could also provide an OTT service to their subscribers in order that they could try to compete with Netflix. The system was designed for small screen Android and IOS devices, i.e. up to 10 inches. We used DRM to encrypt the content.
2. I was responsible for security policy and security architecture. I worked closely with the product management and the S/W development team leader in order to determine security requirements, their costs and benefits, and was the architect of their implementation. In many cases the security features were very complex, so I first needed to create a working proof-of-concept, before finalizing their specifications.
3. Originally the exclusive focus of security was protecting the devices from leaking content and keys, i.e. from being reverse engineered. We relied heavily on the premise that we refused to play on "rooted" Android devices or "jail-broken" IOS devices.
4. But as time went by, rooted Android devices became inexpensive and ubiquitous in the consumer market. Therefore due to declining royalties, the major studios (e.g. Disney, Sony, Warner Bros., etc.) were economically forced to allow playback on rooted devices. Therefore additional security had to implemented on the back-end web servers, e.g. to check whether or not a subscriber downloaded an unusually high number of hours of content, or whether the subscriber had

simultaneous downloads from different IP addresses. I designed a secure and efficient data logging system. We logged data in order to better understand how subscribers were using the system, and in order to detect piracy. With tens of millions of subscribers, we collected a huge amount of data. I worked with data scientists to design “big data” collection and analysis techniques. And there was the economic challenge to minimize the communication costs of the data collection program.

5. I specified the anti-reverse engineering and obfuscation programming frameworks and libraries in C/C++.
6. My typical development methodology was to first build a prototype for desktop Linux, secondly as a standalone CLI application on the target device, and finally to hand over a working prototype to the development team. Whenever possible I preferred to test on virtual machines.
7. The challenges of implementing obfuscation are that (1) the other programmers should not be concerned about it because their focus must be on writing correct code, and (2) the resulting increase in size and reduction in run-time speed must not noticeably reduce the usability/functionality of the application. In general the aim of obfuscation is to provide “good enough security” that will deter 95% of potential attackers, and when combined with regular application updates will force an attacker to begin his next reverse engineering attempt from scratch.
8. All secure code modules on the device were implemented as native libraries written in C/C++. Typically offline utilites were implemented in Python.
9. I developed a lightweight obfuscated cryptographic library implemented as a H file using inline functions so that every module that included it had its own private copy of the library with a module specific randomized implementation which prevented an attack against a single core cryptographic module that could potentially subvert the whole application. Due to performance reasons, there were many cases where we could not afford to use AES, especially during the performance critical movie playback which itself relied upon AES decryption, therefore in these cases we used lightweight algorithms instead, e.g. Xorshift PRNG stream cipher.
10. I developed an Android root detection mechanism using fuzzy logic techniques.
11. I developed a light weight dynamically randomized method to efficiently shroud all system calls so that their address is calculated just-in-time in registers before the call is made. The technique fooled the professional reverse engineering debuggers Hex-Rays and OllyDbg which normally can automatically identify and place anchors on the system calls.
12. I created a prototype of a dynamic shared library (DSO) that formally exported no symbols. In fact it used an asynchronous back channel that allowed the DSO to communicate with its caller by using a function declared with the gcc constructor attribute that executes before dlopen() returns. Typically this technique could be applied to *binary* DSOs from our 3rd party vendors *without their knowledge*, by doing binary object file “surgery” (i.e. editing). We were faced with the challenge of securing a critical 3rd party library, i.e. the video codec, which did not incorporate any obfuscation and where the vendor would not allow us to see or to suggest any modifications to their proprietary source code. They produced a single “one size fits all” library for all of their clients. Without securing this library, it was an ideal attack vector.
13. I developed vector operations for the C preprocessor that allowed a stream cipher to be applied to a constant string *pre-compile* time that was used to shroud function name strings that were dynamically loaded using dlsym().

14. I was responsible for the purchase decisions and usage policy of 3rd party obfuscation and cryptographic utilities and libraries. The two main 3rd party utilities that we used were the InterTrust WhiteCryption SCP obfuscating C/C++ compiler and their SKB “whitebox” cryptographic library.
15. See more details.
16. **At the end of my 6 year tenure there were 40M subscribers, and no security breaches.**

2016-16: Cybersecurity Consultant: Protection of a Small Business with Extremely High Security Concerns

@Anonymous, Jerusalem: See details.

Keys: cybersecurity, privacy, anonymity, WordPress, static web site, Cloudflare, Windows, Android, Google Docs, Google Drive

2010-11: VP R&D: Transportation: Urban Traffic Vehicle Route Guidance Algorithms

@TeleQuest (defunct), Jerusalem: See details.

Keys: urban vehicle route guidance, architect, algorithms, Java, AWS

I designed and implemented algorithms along with a computational infrastructure for urban traffic vehicle route guidance similar to what Waze does today.

2009-09: S/W Architect & Developer: Bioinformatics: Invented Algorithm To Overcome PCR Inhibiition

@Syntezza Molecular Detection (defunct), Jerusalem: See details.

Keys: bioinformatics, PCR, algorithms, architect, mathematical programming, data science, AI, C, Python

2004-09: NDS: Cybersecurity Researcher for a CA Satellite Pay TV System

@Cisco-NDS, Jerusalem

Keys: cybersecurity, DRM, algorithms, cryptography, anti-reverse engineering, obfuscation, LLVM compiler, VM, QEMU, RPC, automated testing, S/W quality, C/C++, TCL, Python, Linux, bash, Win32

1. I worked on a wide variety of security related projects. My background task was to do C/C++ code security reviews. Typically secure coding is achieved by adhering to best programming practices.
2. I was a member of the architecture team for their in-house LLVM obfuscating C/C++ compiler.
3. I developed techniques using Virtual Machine (VM) technology to crack Digital Rights Management (DRM) schemes.
4. I developed a technique to subvert a commonly used class of random number generators (RNG) that are seeded with the X86 CPU clock cycle counter (RDTSC) [2]. Many cryptographic algorithms rely upon an RNG for their initialization “handshake”. I implemented this technique by hacking the open source QEMU VM emulator written in C. This attack enabled subscriber credential sharing.

---

2 Today (2017), it is preferable to use Intel’s built-in H/W RNG instruction RDRAND.

5. I wrote the technical specification for CCTV (Chinese government TV) to secure the TV feed of the 2008 Beijing Olympic broadcasts against international piracy. We almost won the contract, until Microsoft offered to do it for free.
6. I wrote business development reports about applying NDS security technology (1) to protect computer games against hacking, and (2) to protect inkjet printers from using 3rd party cartridges.
7. I arranged for well known security experts to give lectures and seminars at the company. The most successful and well attended course was a one week seminar on reverse engineering X86 assembly code using OllyDbg. The course was given by Kris Kaspersky who today (2017) works at Check Point. Afterwards many of us in the security group spent a number of months challenging each other with anti-reverse engineering riddles. The strongest techniques that I developed were (1) jumping into operands of long instructions that were designed to be short opcodes, (2) dynamic creation of opcodes in the heap, and (3) creation of obfuscated FORTH-like virtual machines.
8. I gave the following 3 well received lectures: (1) how to write code that mitigates side-channel attacks, (2) advanced TCL for the smartcard testing group, and (3) developing a suite of small Posix style utilities that are LEGO-like and easy to interface in unexpected/serendipitous ways versus building large monolithic utilities that are accessible via their GUI only.
9. I architected and implemented a hybrid simulator/emulator debugger written in C for legacy Set-Top Boxes (STB) that originally could be debugged only with printf statements to log files. My new debugger allowed source code on the PC to be debugged using the MS Visual Studio IDE debugger while still viewing the results on the STB. Implementation was accomplished by reverse engineering of the STB middleware API. 80% of the middleware ran natively on the PC, while the STB low level H/W specific portions were implemented via an agent on the STB that was accessed via API calls that were implemented as Remote Procedure Calls (RPC).
10. I wrote an automated testing system in TCL and C/C++ for a satellite content delivery system for huge content, e.g. delivering ultra high definition movies to cinemas, and print newspapers for remote publishing. I created a Domain Specific Language (DSL) in order to execute the satellite operations. After studying the Win32 C/C++ source code of the satellite ground control station, I detected a major conceptual flaw which the architect refused to believe (because testers are not supposed to understand Win32 internals!). So I wrote a progressive test that brought the satellite to its knees at only 25% of its rated capacity. Afterwards the development team used my tool to develop their own unit test scripts, and to execute a system sanity test before checking-in any changes to the source control system.
11. I did a study for senior management by data mining the company's bug database, which showed them that 25% of S/W development manpower was wasted on fixing bugs. And I presented them with simple techniques that could reduce this number by 80%.

2002-03: CTO & S/W Architect: Accessibility: Invented System to Allow Blind to "See" Sonic Maps

@Virtouch (defunct), Jerusalem: See details.

Keys: accessibility, blind, architect, algorithms, GIS, MapML, HTML, SVG, javascript, XSLT, XML Schema, XSLT, C, TCLNDS

1999-2002: Vyyo: S/W Mngr & Architect: Network: Embedded & Offline Utilities for a “Wireless” Cable Modem and Router System

@Vyyo (defunct), Jerusalem

Keys: network, architect, algorithms, SNMP, SNMP-agent, NMS, automated testing, C, TCL, embedded

1. I was the architect of the SNMP network management system (NMS), MIB, and embedded SNMP agent.
2. I was the architect of a hybrid IP connection for cable modems where there was no physical cable upstream channel. Instead the upstream channel used a telephone modem (ATA), while the downstream channel used the cable modem. For typical surfing, the effective downstream rate was as fast as a pure cable solution. The company applied for a provisional patent.
3. I greatly improved the efficiency of the laboratory modem speed stress testing by a factor of 10-100 by using a steepest descent search algorithm instead of a binary search algorithm. Reduced testing time per modem from hours to minutes.
4. I designed a *virtual* testing lab with 64K modems and 512K PCs by multiplexing the *physical* connections. The test lab had only 256 *physical* cable modems, and 4 *physical* PCs with 8 network connections each.
5. I designed a very efficient hash table algorithm in C for the router’s arp table cache, based upon an algorithm I had invented 10 years earlier. The special features of the hash table algorithm were no use of dynamic memory allocations for embedded safety, a unique 2^N table size algorithm that required no use of modulo/division operations for efficiency, and a LIFO queue in order to gracefully handle table overflow.
6. I designed a flash memory file system for the modem and router.
7. I designed a TLV configuration file utility both offline for the PC workstation, and embedded on the modem and router.

1998-99: S/W Architect & Developer: Compiler: GCC Compiler Port for a 128-Core Stack Machine

@Fourfold Technologies (defunct), Jerusalem: See details.

Keys: gcc C compiler, architect, algorithms, DSL, C/C++, FORTH, LISP, TCL

1997-98: S/W Architect & Developer: Factory Automation: Conoscopic Interferometer Workstation

@Newport-Optimet, Jerusalem: See details.

Keys: measurement workstation, architect, algorithms, DSL, C, TCL, OpenGL, Win32, soft real-time

1996-97: Lecturer: Win32 Internals Course

@Mer Group, Jerusalem

Keys: lecturer, Win32, C

1995-96: Elop: CTO & Architect: US DOD Mil-Spec Automated Testing: Night Hawk Fire Control System

@Pitkha Outsourcing (defunct), Jerusalem for @Elbit-Elop, Rechovot: See details.

Keys: automated testing, mil-spec, architect, DSL, C/C++, lex/yacc BASIC compiler, Win32, soft real-time

1995-95: Lecturer: Introductory University Computer Science Course on Database Theory

@Michlala College Bayit Vegan, Jerusalem

Keys: lecturer, database, SQL

1991-94: DSPG: CTO & S/W Architect: VLSI: Simulator & S/W Toolchain For DSPG PINE CPU

@Pitkha Outsourcing (defunct), Jerusalem for @DSP Group, Givat Shmuel: See details.

Keys: VLSI simulator, S/W Development Toochain, architect, algorithms, DSL, C/C++, lex/yacc, assembly, Win32

1989-91: DEC: S/W Architect & Developer: Factory Automation: Shop Floor Production Control (SFPC) System: BARI II

@Digital Equipment Corporation (DEC) (defunct), Herzliya for @Iscar, Tefen: See details.

Keys: factory automation SFPC, architect, algorithms, DSL, Pascal, SQL, VAX/VMS

1988-88: S/W Architect & Developer: Accessibility: Quadriplegic PC Accessibility

@Cubital (defunct), Herzliya - a charity project funded by the company and their CEO Itzhak Pomerantz in cooperation with the Lowenstein Rehabilitation Hospital, and the IDF Rehabilitation Unit: See details.

Keys: accessibility, Prolog, PC-DOS

1987-88: S/W Developer & VAX/VMS Sysadmin: 3D Printer: Solider

@Cubital (defunct), Herzliya: See details.

Keys: 3D printing, C, sysadmin, VAX/VMS

1986-87: S/W Developer: Soft Real-Time RS232 Z80 Communication Driver: Data Collection & Access Control Terminal

@Elde (defunct), Jerusalem

Keys: data collection terminal, C, RS232, Z80, embedded, real-time

1985-86: S/W Developer: Factory Automation: Leather Sewing Workstation

@Orisol, Lod: See details.

Keys: sewing workstation, DSL, algorithms, AutoCad, C, awk, PC-DOS

1984-85: S/W Developer & VAX/VMS Sysadmin: Hebrew/English Word Processor: Glyph

@John Bryce, Jerusalem

Keys: word processor, C, sysadmin, VAX/VMS

1983-84: Elta/IAI: S/W Developer: Real-Time: Data Collection Terminal & Radar for Lavi Fighter Plane

@DSI (defunct), Givatayim for @Elta/IAI, Ashdod: See details.

Keys: data collection terminal, PL/M, 8080, RTOS, fighter plane radar, Jovial, embedded, real-time

1981-83: Mitre Corp: S/W Developer & IBM CP/CMS Assistant Sysadmin

@Mitre Corp, McLean VA: See details.

Keys: APL, PL/1, sysadmin, IBM CP/CMS

1979-80: Programmer & Economist

@JWWA.com, an economic consulting firm in the Washington DC area: See details.

Keys: electric utility economics, Fortran, IBM MVS

1977-78: Intervenor/Economist

@Ontario Energy Board (OEB), Toronto: See details.

Keys: electric utility economics

3.0 Education

3.1 Formal Education

1979: York University, Canada: MA Economics & Applied Mathematics

See details.

1977: University of Toronto - Rotman School of Management (MBA Program): No Degree

See details.

1976: University of Toronto: BA Economics & Applied Mathematics

See details.

3.2 Continuing Education

Today the field of computer science is changing so rapidly that one's formal education has a half-life of less than 5 years. Therefore in order to maintain my state-of-the-art professional edge, I am involved in an intensive effort of continuing education. See details.

4.0 Spoken Languages

1. English (5/5)
2. Hebrew (4/5)
3. French (2/5)

5.0 Computer Languages, SDKs, and Operating Systems

Language knowledge in order of expertise, based upon my current frequency of usage:

1. C, TCL, bash + posix text utilities, e.g. awk, sed, etc.
2. C++, python, make, html5, css, markdown, pandoc, jinja2
3. flex, bison, llvm, javascript, java, yaml, json, go
4. forth, lisp, prolog, apl, fortran, opengl, svg, xml schema, relax ng, xslt, perl, C#

Note that I write compilers and Domain Specific Languages (DSL), so learning a new language takes me only a few days.

O/S knowledge in order of expertise, based upon my current frequency of usage:

1. Linux
2. Android
3. Win32
4. IOS

6.0 Patents Under Development

- Bioinformatics: (a) An extremely accurate and simple noise reduction and normalization algorithm to improve the accuracy of the standard PCR Ct calculation, and (b) an Artificial Intelligence (AI) methodology for measuring the quantity of DNA in a bioassay where inhibition makes it impossible to estimate the Ct because no underlying logistic function (= a flat "S" shaped curve) exists.

- Cryptography: A set of non-linear cryptographic primitives using Hamming weight-like data dependent permutations which overcomes the well known limitation of using Hamming weights because they have a binomial distribution.

7.0 Personal

I was born in Canada in 1956. I have lived in Jerusalem Israel since 1983. I am married with 4 children, 2B + 2G, plus many grandchildren. I take physical fitness seriously. Once upon a time I was a judoka, and a classical guitarist. I was an IDF reserve soldier for 15 years, where I served as a combat soldier in the infantry in the Jordan Valley. In spite of the fact that I joined the army when I was 32 years old (Hebrew: *Shlav Betnik*), functionally, but unofficially, I served in the capacity of deputy company commander (Hebrew: *Samech Mem Pe*) which provided me with the opportunity to achieve rich personal growth, and enabled me to learn important managerial and leadership skills.

Colophon

- **Generator:** This document was generated using the Pandoc universal document converter extended Markdown engine, along with the Jinja2 macro/template preprocessor. See the source code at my github site.