Avraham Bernstein

Expert in cybersecurity, algorithms, domain specific languages. Master S/W Engineer for Industrial Strength Apps.

Avraham.Bernstein@gmail.com

Summary

I am a computer scientist. I am a master S/W engineer and architect of industrial strength systems. After 30+ years I still feel it is the most creative profession that I could have chosen. I work at executive level. I am very innovative and creative. I have a strong background in economics. I communicate well both orally and in writing.

For the past 13 years, I worked primarily in the field of cybersecurity with a specialty in S/W obfuscation, i.e. anti-reverse engineering techniques. The target O/S were Android and IOS.

I have also worked in a broad range of other application domains where I have quickly come up to speed, and functioned at an expert level - because I am self-learner, and I know how to find and utilize mentors who are experts in the new field, and who also recognize my potential. (I am also a good mentor myself). These applications include bioinformatics, transportation, blind vision, VLSI, and factory automation. Just a few years ago, after only a month in the new (for me) field of bioinformatics, I developed a revolutionary new PCR Ct prediction algorithm. I was very fortunate that my mentor was the one who developed the original algorithm - which is the most cited in this field in Google Scholar.

Invariably I am able to apply knowledge, or "meta-knowledge", that I learned in previous fields, and apply it to the new one. A common technique that I use is to create a domain specific language (DSL) that allows me to write a formal specification of a problem. With a little ingenuity the DSL can be used to create a working command line application, and it usually forms the basis of an automated testing framework. I successfully used DSLs to develop a shop floor production control system, and I used it to describe CPU architectures.

A common programming technique that I use is meta-programming with templates that allows me to automatically generate code by creating intelligent macros regardless of the underlying target programming language

Experience

Head of Security: OTT Internet Pay TV "Secure Player" at Viaccess-Orca
June 2014 - May 2017 (2 years 11 months)

During the 6 years of my tenure as head of security for this OTT (over-the-top) Internet pay TV product, we had tens of millions of subscribers, while we were never successfully hacked in the field. The product was licensed for small screen devices, i.e. Android and IOS smart phones and tablets.

My job was to prevent the theft of intellectual property (aka anti-piracy), namely to protect the DRM app that we loaded into subscriber devices against reverse engineering.

Most important, I had to protect the secret keys that were stored on the subscriber devices from leaking, i.e. the subscriber keys and the content decryption keys. The core security module that we loaded into the subscriber devices was a binary (native machine code) library written in C/C++, where just the GUI wrapper was written in Java and Objective-C. Note that it is 10-100 times more difficult to reverse engineer C/C++ code compared to Java or Objective-C code. I used state-of-the-art binary software obfuscation techniques in order to outsmart state-of-the-art reverse engineering assembly code debuggers, i.e. IDA Hex-Rays.

I developed many of the security algorithms including the Android root detection algorithm which needs to be implemented using fuzzy logic.

Our major security tool vendor was Intertrust / WhiteCryption who supplied us with their SCP obfuscating compiler, and their SKB "whitebox" cryptography SDK. I worked very closely with the vendor to improve their product, because I was previously involved in the specification of a LLVM obfuscating compiler at NDS.

From a managerial perspective there were critical economic decisions that had to regularly be made. Overly aggressive piracy detection could generate thousands of customer support calls from bona fide subscribers where the cost of handling an individual call could be more the price of the monthly subscription fee. But weak detection algorithms could result in content piracy, or counterfeiting of subscriber credentials.

Head of Security: OTT Internet Pay TV "Secure Player" at Sansa Security (ARM)

March 2011 - May 2014 (3 years 2 months)

Viaccess-Orca purchased the "Secure Player" project and the development team from Discretix - the original name of Sansa Security - so see the full description immediately above under my work at Viaccess-Orca.

VP R&D: Urban Traffic Route Guidance

January 2010 - February 2011 (1 year 1 month)

TeleQuest was a startup engaged in the development of a coordinated dynamic route guidance system, in order to alleviate traffic congestion in large metropolitan areas - similar to what Waze now does.

Simulation experiments on Tokyo traffic showed that at a guided vehicle penetration rate of 50%, TeleQuest could provide a sustained average travel time reduction of ~50% for guided vehicles and ~40% for non-guided vehicles. None of the competing commercial or academic models come close, because after penetration rates of 15-20% their benefits peak and then started to decline because their simplistic opportunistic algorithms generate their own secondary traffic jams of the guided vehicles.

My job was to architect, develop, and implement algorithms, in conjunction with a team of transportation scientists and mathematicians, that were executed in a real-time traffic simulator running on tens of cloud computers, to be coordinated with GPS equipped smartphones running our S/W on tens of thousands of vehicles.

Even though it was beyond the scope of my work, I learned that the single most important factor for improving the flow of traffic in congested metropolitan areas is to prevent "hunting" for parking spots. IBM is working on this problem in their "smarter parking" project, which is a part of their larger "smarter cities" initiative.

Consultant: Bioinformatics PCR Algorithm Development

September 2009 - December 2009 (3 months)

Within 3 months of entering the field of bioinformatics, I discovered a new much more accurate PCR (DNA amplification) Ct prediction algorithm that worked particularly well in noisy environments due to PCR inhibition. I was working for a company developing a new dry MRSA detection kit, where MRSA is a deadly strain of antibiotic resistant staphylococcus bacteria that thrives in hospitals, and is the number one cause of fatal infections in hospitals. When I arrived their kit was failing. Its detection rate was only 50%, and as a result the company was facing liquidation because the investors were losing confidence. I raised their detection rate to 95+% (!) which was about 10% higher than the detection rate of their competitors - mainly from the giant multinational pharmaceutical companies. Before beginning this project, I had zero background in bioinformatics, biology, or biochemistry.

During this short period I made some important discoveries:

1. I discovered a much more powerful and much simpler algorithm to normalize data, and to reduce noise - compared to the classic Bar-Tichopad algorithm. Note that Tzachi Bar was my mentor.

2. I discovered an alternative AI (artificial intelligence) based algorithm that very accurately determined whether a sample is positive or negative (according to the Petri dish results) when the underlying data did not exhibit the flat "S" shape expected by the Bar-Tichopad algorithm.

I am in the process of patenting my algorithms.

Security Research & Business Development at NDS service Pay TV Technology

March 2004 - September 2009 (5 years 6 months)

1. Background task was doing security code reviews. At a basic level this involved enforcing the use of textbook safe programming practices, e.g. protection against stack smashing, wiping memory before freeing

it, validating user input, etc. At a higher level, it involved checking how well the program was protected against reverse engineering.

2. We came to the conclusion it was far too difficult for the average programmer to be concerned about both producing correct code and to protect it against reverse engineering. I was part of the specification team for an obfuscating compiler, to be implemented using the open source LLVM infrastructure.
3. I was technical lead and writer of a proposal to China CCC TV to protect their internal IP-TV broadcasts of the 2008 Beijing Olympics from being leaked outside of China. The IOC allows the host country to receive the feed for free, but the owner of the feed MSNBC received billions of dollars of royalties. Therefore if China were incapable of protecting the feed from leaking, MSNBC would cut off their internal feed.
4. I wrote business proposals to use NDS security technology to protect computer games and to protect printer ink cartridges from reverse engineering.
5. I became an expert in the use of virtual machines (VM) technology to crack DRM and cryptographic security schemes. I used QEMU to crack crypto techniques developed by the world's best commercial crypto team (at NDS) by breaking the random number generator.
6. I arranged for seminars to be given by world class hackers and security researchers.
7. I gave a presentation to senior management about the huge development costs associated with fixing bugs. Nearly 25% of their R&D manpower was devoted to fixing bugs! I discovered the extent of these costs by "data mining" their ClearQuest bug tracking database. And I also suggested numerous relatively easy to implement techniques that would greatly reduce the number of bugs.

CTO: Blind Accessibility H/W & S/W To Enable Viewing of Digital Images & Maps
January 2003 - March 2004 (1 year 2 months)

1. The flagship product was the VTPlayer tactile mouse. It had 2 finger pads on top - each with an array of 4x4 refreshable braille pins.
2. In default mode, it converted screen pixels to an 8-bit gray scale with say a threshold of 128 to control whether a pin is up or down. By scanning the mouse across the screen every pixel could be felt.
3. The problem was that blind people are not comfortable with mice because they can't see the cursor. I added vibrating pin patterns and tonal audio cues to indicate the cursor position. However the solution was not good enough, and anyway the custom mouse was extremely expensive (~$350) given our low sales volume.
4. Therefore I abandoned the mouse in favor of an off-the-shelf graphics tablet and stylus combined with audio cues. A tablet computer or smart phone are also good.
5. It was a great success because the blind could inherently feel where their position was on the tablet. And we rectangularly mapped the tablet into the physical screen or the current window.
6. XY position audio cues were based upon 2 musical instruments, e.g. piano and flute, mapped to 8 octaves of a major scale. Each of the 56x56 quadrants (note a scale has 7 notes) produced a different note combination that they could easily discern. The 8-bit gray value of the underlying pixel was indicated by 256 levels of volume.
7. My most important conceptual breakthrough was using XML and SVG to provide semantic/contextual information, e.g. via MapML. Maps today are made of layers, e.g. roads, cities, sites of interests, topography,

polical borders, etc. I provided filters to declutter the maps so only specified layers would be selected. Also the granularity was selectable too, e.g. topographic isolines of 500m. By clicking the user received contextual semantic data associated with the cursor position.
8. I was awarded a large EU FP6 matching grant to conduct further research - administered the Israeli gov Chief Scientist office - but the investors balked.

Senior Programmer: For Cable Modem & Router
January 2002 - December 2002 (11 months)

Manager Cable Network Tools S/W Development Group at Vyyo
January 1999 - December 2001 (2 years 11 months)

1. I managed a team of 3-6 programmers.
2. I designed a NMS and MIB for cable modems and routers, and I designed the associated embedded SNMP agents.
3. I designed a TLV (type-length-value) configuration file system, both the offline GUI and CLI, and the embedded interface inside the modems and routers.
4. I designed a dual boot flash file system on the embedded side.
5. I designed an efficient embedded FIFO hash table algorithm used to implement the router's arp table.
6. I designed 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. Head end network equipment for both interfaces was supplied by the cable operator. My solution was to dynamically modify the arp table of the edge router. For typical surfing, the effective downstream rate was as fast as a pure cable solution. The company applied for a patent.
7. 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.
8. Designed a virtual testing lab with 64K modems and 512K PCs via multiplexing the physical connections. The test lab had only 256 physical cable modems, 4 physical PCs with 8 network connections each, 1 cable router, and 2 24-port layer-2 programmable switches. By dynamically editing the PC MAC addresses, and by dynamically editing the MAC filters on the network switches, I was able to stress test the router into believing that it faced 64K modems, and the modems into believing that each one was shared by 16 PCs.
9. Designed an embedded logger that could be dynamically configured for selecting the messages to be emitted. Further reduced message traffic by using message IDs to remove static text strings from messages.

Consultant: JOLT: Network Management System for FSO Devices at MRV Communications
January 2000 - December 2000 (11 months)

1. The FSO (free space optics) repeaters enable fiber optic cables to be extended by kilometers through the air.
2. The devices were made of digital H/W with no need for a CPU.

3. In order to provide network management capability, I selected an inexpensive micro controller that could interface with the FSO H/W, had an Ethernet port, and a built-in Java interpreter.
4. I designed the MIB, and an embedded program that could interface with the H/W.
5. I designed a simple HTTP server that acted as an SNMP proxy agent.

Consultant: Network Driver Software at Inex/Zamir
January 1999 - December 1999 (11 months)

The company products, license plate recognition for parking and toll roads, were written in Visual Basic (VB) for Windows NT PCs. They required asynchronous UDP network communication capability, and also required asynchronous FTP capability, neither of which were supported by the VB API. I designed and implemented these Win32 network interfaces for them in C as a DLL with an API that was accessible by VB.

Senior Programmer: GCC Compiler Port for a 128-Core Stack Machine
January 1998 - December 1998 (11 months)

This was a very challenging gcc port (using the GNU Compiler Collection framework) because the architecture had no registers (or alternatively an infinite number of registers) while RAM access was highly unusual in order to accommodate the 128 cores. I needed to add pragmas for co-processing. The machine instruction set was FORTH-like, so it presented some unusual optimization challenges. At the end of the day, the C compiler worked and produced efficient code.

I noted early in the project that the method for emitting target object code was well suited to an object oriented design. At the time the gcc compiler collection only interfaced with C code. In order to lower the "impedance" of using C++ it was possible to create a static wrapper for the C++ classes that were acceptable to a C project.

The final source code was extremely repetitive. Therefore I developed a preprocessor in TCL in order to automatically generate much of the code.

Contractor: S/W Architect & Implementation of Interferometer Workstation at Optimet Ltd
January 1996 - December 1997 (1 year 11 months)

1. The company product was a measurement workstation based upon their proprietary conoscopic probe.
2. The object to be measured was on a platter that could be moved in the XY axes.
3. The probe was static, and measured Z - actually an array of Z points.
4. The workstation also included a video camera.
5. Generally the probe could scan at its maximum speed, although some types of materials, especially reflective, required lower speed.
6. My task was to design the S/W architecture for a Windows NT PC to implement an *automated* workstation that could be used in a factory environment.
7. I created a domain specific language (DSL) in TCL for controlling and configuring the workstation.

8. The most interesting and challenging aspect of the project was to give "life", i.e. semantic meaning, to the millions of raw XYZ data points that were measured.
9. I used OpenGL for this purpose.
10. I created a toolkit and domain specific language for the 3D visualizations.
11. My reference development object was a tooth (aka "Timmy the Tooth") which I painted as if it were a topographic map.
12. An important visualization technique for mass produced widgets on the factory floor, was to compare them against a reference widget.

CEO & CTO: Architect & Implementation of US DOD Mil-Spec Automated Testing System
July 1994 - December 1995 (1 year 5 months)

1. The company product was the weapon fire control system for the Night Hawk laser guided missiles.
2. My task was to create an architecture and implementation of the automated test procedures dictated by the US DOD. The test specification documentation was about 1 meter high.
3. Additionally there would be many ad hoc test that would be required during the development process.
4. The test equipment included external voltage regulators, external heating and cooling equipment, vibrators, etc.
5. The workstation controlling the tests was a Windows NT PC.
6. Instead of creating a monolithic test program in C/C++, I created a BASIC-like domain specific language (DSL) with special drivers for controlling the various pieces of H/W.
7. The test specification manual was being regularly revised, plus I needed the flexibility to allow the system engineers to write their own scripts without delving into the C code. And I wanted to avoid having the client call me every time he needed to implement a minor/trivial change that he could easily learn to do himself.
8. The major tests were presented via a GUI which in fact emitted DSL script.
9. The system worked as planned.
10. Unsupervised tests ran successfully for up to 72 hours (over holidays).
11. Post mortem: The first version of TCL was released around the time of this project *pre -Internet*. Once I learned about TCL, I realized it would have been the ideal platform for creating the test environment. Instead of me having to create my own BASIC-like language with control structures and variable handling, I could have relied upon TCL, and simply added custom primitives for the various pieces of H/W.

CEO & CTO: Architect & Implementation of S/W Toolchain For DSPG PINE CPU
January 1991 - June 1994 (3 years 5 months)

I was the software architect of a clock accurate DSP CPU simulator along with a complete software development toolchain, i.e. a debugger, C compiler, assembler and linker. Note that the system was developed just before the GNU Compiler Collection framework reached maturity (i.e. v2.95).

This system enabled complete working applications to be developed *before* the chip became physically available. This system reduced the client's time-to-market by 6-12 months. The technological breakthrough was my design of a domain specific language (DSL) that described the CPU architecture which automatically generated the source code for the toolchain that enabled the system to be automatically rebuilt within an hour in the face of almost daily changes to the VLSI architecture - especially the pipeline.

Another important application for this toolchain was to compare the results of assembly code test programs used to verify the VHDL VLSI specification against the same test code run on the simulator. The simulator executed tests for up to 72 hours (over holidays) without any failures.

Architect: Shop Floor Production Control (SFPC) System "BARI II"

January 1989 - December 1990 (1 year 11 months)

1. Iskar Matkash in Tefen IL is a fully automated factory that produces thousands of different cutting blades using a sintering process. The raw materials go through many stages of operations. In many cases after undergoing intermediate processing, the partially processed material can still be diverted to multiple final products - similar to stem cells. The factory contains hundreds of automated workstations, stands, stacks, guided vehicles, and conveyor belts. Pallets can be automatically move from one stand on a workstation to the next, or to temporarily place a pallet on a storage stand or a stack stand.
2. My task was to create a computer program that automatically operated/orchestrated the factory.
3. When my co-architect and I started this project we had zero background in industrial engineering. We were supplied with a mentor who brought us up to speed.
4. Eventually after months of discussions we created an architecture that was a textbook object oriented taxonomy - a "factory object kingdom". The top level object was a "production instruction".
5. We defined the attributes and methods associated with each object.
6. We created a descriptive, i.e. *non-procedural*, domain specific language (DSL) that was designed to be used by the factory engineer.
7. I wrote the language manual.
8. We used the language to configure the factory. We created a GUI interface wrapper for the language which emitted CLI script.
9. We mapped the language to a relational database.
10. We created an implementation architecture in Pascal. Given the inherent OO nature of the architecture, C ++ would have been a better implementation language choice - but the project management refused.
11. After 18 calendar months, and 4 man-years later, the factory ran perfectly!

S/W Architect & Implementation: Quadriplegic PC Accessibility

July 1988 - December 1988 (5 months)

I designed and implemented a system that enabled quadriplegics to access computers with CRT video displays *before the era of inexpensive voice recognition technology, tablets, and smartphones*.

The hardware we used for this project was a standard light-pen (an obsolete point-and-select input device that determines its position via the CRT's scan point) combined with custom telescopic optics that extended its normal maximum range to the screen from 10 mm to 800 mm, which we attached to the user's head with a headband, along with a standard accessibility sip-and-puff switch.

The software had 2 modes of operation. In learning mode, a virtual keyboard covered about 80% of the screen. (With 6/6 hindsight, I should have patented the virtual keyboard concept !). When the light-pen was aimed at a key then it would be illuminated. Puffing on an illuminated key caused the virtual key stroke to be entered into the operating system. In normal mode, the virtual keyboard was no longer visible. Instead a single key would pop up at the position on the underlying hidden virtual keyboard where the light pen was aimed, leaving 95% of the screen real estate available for the "real" application.

This system enabled our first client, Shulamit Gabai, a former school teacher who was stricken with polio in all of her limbs, to type 30 characters per minute, and to become a book editor for a major publishing house.

I implemented the system in the Prolog programming language, just in order to learn the language.

Senior Programmer: 3D Printer "Solider"

January 1987 - December 1988 (1 year 11 months)

1. This was a 1st generation 3D printer. Even though it was the size of a 6x3m shipping container, it relied upon the same principles that the modern printers use. Each layer was made of polymer and wax, where the wax was melted and flowed out once the object was complete. One of my most important programming tasks was writing a VAX/VMS printer driver in C where each layer was a logical page.
2. I implemented the GUI using X-Windows "Motif".
3. I maintained the source control system.
4. I was the sysadmin for their VAX/VMS system.

Programmer: Automated Leather Sewing Workstation at Orisol Asia Ltd.

January 1986 - December 1986 (11 months)

This was my first project where I began using domain specific languages (DSL).

The challenge of this system was due to the fact that no two pieces of leather are identical because it is a natural material. Therefore it was not possible to blindly sew according to the pattern. Real-time video feedback was required to control the workstation. And given the high torque resulting from the extremely high speed of the needle, the needle speed had to be carefully reduced - especially when making turns. The patterns were created using AutoCad.

I added "attributes" to the AutoCad pattern specification, similar to road signs used on streets and highways. The basic attributes were used to control the operation of the sewing machine. The video attributes were used to give hints to the live image processing algorithm about what types of problems to anticipate, and what types of filters to use according to the color and texture of the leather. Afterwards I wrote a parser that converted these attributes to the C code used to control the workstation.

Programmer: Hebrew/English Word Processor "Glyph"

February 1985 - December 1985 (10 months)

1. Involved in various C programming tasks for the word processor.
2. I wrote a graphics driver for a smart terminal that supported graphics primitives.
3. I was the sysadmin for their VAX/VMS computer system.
4. I provided telephone VAX sysadmin support to many of their customers who more often than not faced sysadmin problems rather than problems specific to the word processor.
5. These support calls revealed the existence of tens of illegal copies of the S/W often in government agencies - which invariably resulted in proper purchases.

Programmer: Real-Time

May 1983 - January 1985 (1 year 8 months)

1. While waiting for my security clearance, I worked on a data collection terminal. I developed a real-time operating system kernel for an 8080 CPU.
2. After receiving my clearance, I was in charge of the executive task of the radar used in IAI's Lavi fighter plane.

Programmer at MITRE

January 1981 - March 1983 (2 years 2 months)

1. Implemented a computer model for the process of geothermal energy extraction. Relied both on my programming and economic expertise.
2. I was a junior sysadmin for their IBM CP/CMS system. Developed many applications in APL.

Programmer/Economist

July 1979 - December 1980 (1 year 5 months)

I did simulations of electric power generation and transmission systems where the results were presented at Public Utility Commission rate hearings. Relied both on my programming and economic expertise.

Intervenor/Economist at Ontario Energy Board

March 1977 - August 1978 (1 year 5 months)

ECAP'77 Ontario Hydro Costing and Pricing Hearings

I presented Ontario Hydro's marginal cost pricing (aka peak load or time-of-day pricing) case which the government of Ontario forced them to withdraw due to political pressure from the metal refining industry which at that time was using electric blast furnaces. I received extensive "secret" support from the disgruntled economists who wrote the submission. Even though the Energy Board eventually refused to accept the principle of marginal cost pricing, I was the first public interest "intervenor" to ever be awarded costs. And I published an op-ed describing the issues at the hearing in the Globe and Mail - Canada's newspaper of record.

Education

York University

MA, Economics, 1978 - 1979

University of Toronto - Rotman School of Management

no degree, Business Administration and Management, General, 1976 - 1977

University of Toronto

BA, economics, mathematics, 1973 - 1976

Avraham Bernstein

Expert in cybersecurity, algorithms, domain specific languages. Master S/W Engineer for Industrial Strength Apps.

Avraham.Bernstein@gmail.com

Contact Avraham on LinkedIn