Résumé: Evan A. Sultanik, Ph.D.

November 29th, 2017

evan@sultanik.com 240 A Race St., Philadelphia, PA 19106 +1 215 919 7234
https://sultanik.com/ https://github.com/ESultanik https://twitter.com/ESultanik https://keybase.io/ESultanik

CLEARANCE: Information Available Upon Request

2017–Present	Trail of Bits	Principal Computer Security Researcher Advanced Concepts Research Group Lead
2012–2017	Digital Operatives	Chief Scientist
2010–2012	The Johns Hopkins University APL	Senior Research Scientist
2006–2015	Drexel University Department of Comp. Sci.	Adjunct Faculty
2001–2010	Drexel University Department of Comp. Sci.	Research Fellow
1998–2003	Feith Systems and Software	Software Engineer

The following is a chronological account of some of the things I've done and software I've created. I am legally constrained about talking about portions of my work (*e.g.*, proprietary code assessments), so much of it is omitted here. In lieu of a traditional laundry list of technologies and tools with which I'm familiar, I hope the following will serve to demonstrate that I am a generalist with strong computer science fundamentals and the ability to quickly learn and master new technologies. I specialize in computer security, program analysis, AI/ML/NLP, combinatorial optimization, and distributed systems. You might also note that I've unabashedly written this informally. Do you trust people who speak about themselves in the third person? I don't. For a more traditional résumé, my formal academic CV is available on my website or by unzipping this PDF[1].

August, 2017 Edited and co-published *PoC‖GTFO,* a nearly 800 page print collection gathering articles published in the first ten issues of the International Journal of Proof-of-Concept or GTFO. PoC‖GTFO—on which I have been an editor since August of 2015 and a contributor since June of 2014—follows in the tradition of *Phrack* and *Uninformed* by publishing on the subjects of offensive security research, reverse engineering, and file format internals. Until now, the journal has only been available online or printed and distributed for free at conferences worldwide. ISBN-13: 978-1-59327-880-9 https://www.nostarch.com/gtfo

October, 2015–August, 2017 Member of the "control team" of expert *human* vulnerability researchers for the DARPA VET program, the purpose of which was to act as a baseline against which to compare the *automated* vulnerability analysis tools of the other performers. This consisted of using state-of-the-art publicly available, commercial-, and government-off-the-shelf tools (*e.g.*, IDA) to determine whether a given set of black-box ARM binaries had been modified to include malice. Our team outperformed the automated equivalents across four judged engagements. Primarily IDA, HexRays, and our own static analysis tools that I helped develop (see below).

June, 2017 Published and presented an analysis of zoning and residential density changes in Philadelphia over the previous five years. This required devising and implementing a solution to the difficult computational geometry problem of intersecting the tens of thousands of latitude/longitude polygons describing Philadelphia's zoning districts both pre and post the comprehensive zoning overhaul that occurred in 2012. Also used Machine Learning to create a model to predict the real estate tax revenue implications of zoning changes. Implemented in Python, Java, and KML. https://www.sultanik.com/blog/ZoningDensity

June, 2017 Created a PDF that is also a valid Git repository containing its own LaTeX source code and a copy of itself[2]. Published in PoC‖GTFO Issue 0x15. Implemented in Bash, LaTeX, Python, and C. https://github.com/ESultanik/PDFGitPolyglot

June, 2017 Created a toy web-based terminal emulator and shell, backed by a filesystem served over HTTP (similar to WebDAV), in which scripts are dynamically loaded from the server via JavaScript, and replete with libraries like `ncurses`. Implemented purely in JavaScript using JQuery. https://www.sultanik.com/#term

March, 2017 Created a PDF that is also a valid Nintendo Entertainment System ROM image[3] that, when played/emulated, renders the title page of the PDF *and* displays the MD5 checksum of itself[5]. This MD5 quine is achieved by solving for 128 MD5 collisions using an implementation of an algorithm created by Marc Stevens. This polyglot was in collaboration with Evan Teran and Ange Albertini. Implemented in a mixture of C, 6502 assembly, and Python. https://www.sultanik.com/pocorgtfo/#0x14

August, 2013–November, 2016 Created a semi-automated static analysis tool for human-in-the-loop reverse engineering and vulnerability research along with four colleagues. Led the effort in researching and implementing components for static taint analysis, and devised novel algorithms for SMT constraint optimization and detecting algorithmic complexity defects. Implemented in C++11.

October, 2016 Discovered a vulnerability in the way most US defense contractors choose passwords for the spinlocks they are required to use to secure safes and rooms containing classified information. I wrote a short closet drama about it. Implemented in Bash, Python, and English Prose. https://archive.org/stream/pocorgtfo13#page/n42/mode/1up

October, 2016 Created a PDF that is also a valid PostScript file. If you send the raw PDF to a printer, it will print out differently than how it was rendered on screen in the PDF viewer. For good measure, the PostScript also reads your `/etc/passwd` file. Implemented in Python and PostScript. https://www.sultanik.com/pocorgtfo/#0x13

---

[1] `unzip ESultanikResume.pdf ESultanikResume/ESultanikCV.pdf`

[2] `unzip ESultanikResume.pdf ESultanikResume/PDFGitPolyglot.pdf`

[3] This PDF is also a valid Nintendo Entertainment System ROM[4]. You can emulate it in your web browser, here: sultanik.com/nesresume.

[4] By the way, the NES ROM version of this PDF also prints out its MD5.

[5] By the way, the MD5 hash of this PDF is `314159238B65FBF969EC9C4915F0297F`.[6]

[6] By the way, the value of this MD5 hash is not a coincidence, either.

August, 2016 Along with three collaborators, created an AI bot capable of playing Pokémon Go. Unlike other bots of the time which exploited bugs in Niantic's implementation to achieve superhuman feats like teleportation to increase their speed, the goal of our bot was to mimic the behavior of a human player. To our knowledge, it was the first bot capable of battling against humans at gyms. My major contribution was a novel approximation algorithm for a variant of the Euclidean Traveling Salesman Problem (TSP), known as *The Close Enough TSP*, in which the tour does not need to pass through each vertex, but instead simply needs to pass within at most distance δ from each vertex. This was used to find an optimal route for our bot to travel between PokéStops. Implemented in Python, HTML, and JavaScript (the latter for a web-based UI to control the bot).

March, 2016 Created a PDF that is an HTTP quine: It is also a valid Ruby script that, when run, starts a webserver that serves copies of itself. In addition, the PDF is also a valid ZIP, and the Ruby script extracts itself to serve images from the ZIP. Finally, the PDF is also a valid HTML file which can be loaded in a web browser simply by changing the file extension. Implemented in a mixture of Python, Ruby, HTML, JavaScript, and CSS. `https://www.sultanik.com/pocorgtfo/#0x11`

December, 2015–February, 2016 Wrote an application to lift 32-bit ARMv7 assembly into LLVM's intermediate representation. Implemented in Java 8 and LLVM/IR.

June, 2012–October, 2015 Participated as Digital Operatives' Principal Investigator on a $60M+ DARPA program to create a development environment in which an inexperienced programmer could quickly create complex applications by gluing together software components automatically reverse engineered and extracted from existing black-box binaries. I led a team that created a compiler for a subset of Python that produces the resulting application binary. A vital component of the compiler was the creation and implementation of a novel AI planning algorithm capable of runtime contingency planning. Implemented in C, C++, and Python.

June, 2014 Created Lenticrypt: A Provably Plausibly Deniable Cryptosystem. Unlike alternative plausibly deniable cryptosystems like the recently discontinued TrueCrypt—whose ciphertext size grows in proportion to the number of plaintexts (*i.e.,* hidden volumes) it encrypts—Lenticrypt's ciphertext size is proportional to the largest plaintext it encrypts. This is because Lenticrypt shares bytes in the cyphertext between each of the plaintexts it encrypts; they are not stored in separate regions of the ciphertext. Therefore, there is no straightforward way to estimate the number of plaintexts that are "hidden" inside a single ciphertext. In fact, Lenticrypt has the theoretical property that, under reasonable assumptions, there is always a near 100% probability that there exists a key in the public domain that will decrypt a given ciphertext to *any* desired plaintext, even if that key is not known. Published in PoC‖GTFO and popular on GitHub. Implemented in Python. `https://www.sultanik.com/blog/lenticrypt`

May, 2014 Created a custom framework with an incredibly responsive in-memory database for serving my personal website. Implemented in Python, built atop Flask, Whoosh, Bootstrap, and JQuery. `https://www.sultanik.com/`

February, 2014 For most intractable, NP-hard problems, a polynomial time approximation algorithm that can achieve a solution that is no worse than two times the cost of the optimal solution is usually considered good. I created a formal proof that randomly selected feasible solutions to these problems will on average be at least as good as those complex approximation algorithms. I wrote a paper on this, which made it to the top of Hacker News! `http://arxiv.org/abs/1402.0423`

January, 2014 Discovered an efficient way to generate *Isolated Complementary Binary Linear Block Codes*, which Dominic Spill and Michael Ossman proposed as a form of what they call "Unambiguous Encapsulation" to defend against the *Packet in Packet* attacks of Goodspeed, *et al.* Implemented in Python and Satisfiability Modulo Theories (SMT), and ultimately merged upstream into their project. `https://www.sultanik.com/blog/UnambiguousEncapsulation`

October, 2013 Created hash collisions for a number of insecure hashing algorithms used by malware shellcode, including those produced by Metasploit. Michael Myers used this to thwart malware's ability to bootstrap itself after injection. Implemented in a mixture of C, Python, and Satisfiability Modulo Theories (SMT). `https://archive.org/stream/pocorgtfo12#page/n56/mode/1up`

April, 2012–December, 2015 Collaborated with Ali Shokoufandeh and Dario Salvucci to create a novel semantic search engine by ingesting dumps of Wikipedia and learning relationships between articles, words, and images using Contextual Semantic Analysis. This also led to algorithms for cross-language & cross-domain translation, sentiment analysis, and automated text summarization. Implemented in Python, Java, and Lucene.

September, 2011 Created an automated, completely unsupervised, language-agnostic technique for detecting *social signals* in Twitter data. Social signals were discovered by Fink and Montemayor as a phenomena in which certain terms or phrases spike in frequency at the onset of a crisis event. These include "help," "please," and "blood," as well as various profanities, but vary based on the type of event. When I was first briefed on this phenomenon I was immediately intrigued, primarily around the potential of publishing an academic paper peppered with legitimate use of profanity[7]. The paper we ultimately wrote was nominated for the best paper award at AAAI 2012, but ultimately rejected because of "Reviewer #2." `https://www.sultanik.com/blog/SocialSignals`

While a researcher at The Johns Hopkins APL, I had the opportunity to work on a number of different projects, including inferring geotags for social media posts, Linux kernel integrity monitoring, radar coordination algorithms for Naval ships, UAV coordination protocols, and streaming algorithms.

Prior to Johns Hopkins I was at university. I did my undergrad (two separate BS degrees, one in CS and one in Math) concurrently with an MS in CS with a thesis on distributed AI in peer-to-peer networks, followed by a Ph.D. in CS with a dissertation on distributed combinatorial optimization. That was from 2001 through 2010. My time at university was funded

[7] *"... Nam castum esse decet pium poetam ipsum, versiculos nihil necessest; ..."* (For it's proper for a devoted poet to be moral himself, [but] in no way is it necessary for his poems.)

through concurrent work for various government agencies, including the US Army CERDEC, RDECOM, DARPA, and the NRL. I have authored over three dozen academic publications.

In a life previous to that, I was a software engineer at Feith Systems, working on document databases and OCR.