Interview with Gordon Lyon

RIK FARROW

Fyodor Vaskovich (known to his family as Gordon Lyon) authored the open source Nmap Security Scanner in 1997 and continues to coordinate its development. He also maintains the seclists.org, insecure.org, sectools.org, secwiki.org, and nmap.org security resource sites and has authored seminal papers on remote operating system detection and stealth port scanning. He is a founding member of the Honeynet Project, former President of Computer Professionals for Social Responsibility (CPSR), and Technical Advisory Board member for Qualys and AlienVault. He also authored or co-authored the books *Nmap Network Scanning, Know Your Enemy: Honeynets,* and *Stealing the Network: How to Own a Continent.* fyodor@nmap.org

Rik Farrow is the editor of *;login:*. rik@usenix.org

I've known Gordon Lyon, as Fyodor Vaskovich, for over 15 years. Most of our meetings have been online, but we've also attended some meetups in between BlackHat and DefCon.

Fyodor's claim to fame is that he has been working on a network scanning tool, Nmap (nmap.org) longer than I've known him. What began as improvements on the handful of scanning tools available in the '90s has become an open source project as well as a successfully funded business. If you've never used Nmap, it is both a security tool and one useful to network and system admins: Nmap quickly scans networks in a long list of different ways.

There really isn't much on the Web about Fyodor, and we are fortunate that he has taken the time to tell us a bit more about his background and how various parts of Nmap came to be.

Rik Farrow: Can you tell us when you began programming?

Fyodor Vaskovich: I was lucky enough to grow up in a household with computers, since my father had previously worked at IBM. We had systems like the Commodore VIC-20 and the Apple IIe around 1980. And then later we upgraded to an IBM XT clone. In many ways these systems were all less powerful than what we now use in thermostats and watches. But interacting with them and exploring how they worked really taught me a lot.

Eventually I bought my own computer, a 286 running at a whopping 12 MHz! But the best part was the 2400 bps modem! It opened a whole new world of communicating with other computer users on BBSes. Not all over the world, unfortunately, since I couldn't afford long distance telephone calls. But all over the Phoenix metro area, where I lived in those days, was a good start!

Interacting with BBS software was interesting and fun, but I was blown away when I was introduced to my first UNIX shell. It was just so powerful, and the system running this shell had such a fast connection to the fledgling Internet! My high school friend had an account there, too, and raiding each other's accounts taught us a lot about UNIX permissions and security.

I started college studying molecular and cellular biology, but it didn't take me long to decide that computers (especially networking and security) were my real passions, and I switched to computer science. So the tl;dr answer to your question is that I was a self-taught programmer at first, but my university CS education took me to a much higher level.

RF: Until Nmap came out, there were three scanners that I know of: SATAN, very limited and more of a vuln scanner; the one written by Prof (better known as Julian Assange); and part of Hobbit's Netcat (nc). Were any of these early, primitive programs an influence on the first version of Nmap?

FV: Good memory! Yes, I had a full directory of port scanners at that time including the ones you mentioned, plus a very simple one named `reflscan`. Assange's `strobe` was definitely the fastest and most advanced at the time, and I learned a lot from it about nonblocking sockets, rate limiting, and congestion control. But it didn't support options such as the more stealthy

SYN scan or UDP protocol scanning. So I ended up hacking most of these scanners to add new options and features, but in the end I decided it was better to just write my own scanner containing everything I wanted in one program.

I wrote Nmap for my own network discovery use, but then decided to publish it in *Phrack* just in case anyone else found it useful. Apparently they did, and I was inundated with bug reports, fixes, and ideas. So I decided to release one more version. It continued to snowball, but I never could have imagined that I'd still be at it 19 years later as my full-time job!

RF: OS identification is an even earlier feature of Nmap than scripting. I'd become vaguely aware of some obvious differences between operating systems: for example, time-to-live (TTL) values falling into three well-defined buckets. But Nmap looks at a lot more when asked to identify an operating system. Can you tell us how OS identification got started and a bit about the features measured?

FV: Great question! I released Nmap in 1997 as a relatively simple tool for host discovery and port scanning, but then I became increasingly fascinated by the use of TCP/IP stack fingerprinting for the purposes of detecting the operating system running on a remote host. The idea is to send a series of probes to the host's IP address and study the results very carefully for patterns that indicate a certain OS. A tool named Queso really inspired me in this regard, and I tried to take it to the next level with Nmap. I first released Nmap IPv4 OS detection in 1998 and since then it has grown to test dozens of characteristics, from the way initial sequence numbers are generated to the TCP flags used in responses to unexpected packets. TCP options also disclose a wealth of information. Not just whether the remote system supports an option, but also what values it selects for options such as window scale or maximum segment size and the order in which it chooses to list the options in the TCP header.

Our traditional IPv4 OS detection system requires experts to generate signatures of system behaviors based on thousands of fingerprint submissions by Nmap users. This manual system has served us well, but I'm particularly excited about our new IPv6 OS detection system, which uses a very different technique. Instead of requiring experts to write signatures based on specific tests, our new system just feeds all the header data to machine learning systems and lets the computer do the hard work of finding patterns and characteristics of different operating systems in order to match a test subject system with our reference database of known OS responses. Right now we use a machine-learning classification technique called linear regression, but we had an intern this summer named Prabhjyot Singh who worked with a couple of great mentors in the Nmap Project to convert that to a random forest classifier. The results have so far been quite promising, and he continues to improve and test the system even though his Google-sponsored internship is over. We're hoping to integrate it into an Nmap release this year.

I tried to describe this as well as I could in two paragraphs, but folks wanting the full details can refer to our extensive online documentation (https://nmap.org/book/osdetect.html).

RF: Can you tell us about adding scripting capabilities to Nmap?

FV: Sometimes it seems like every program grows and grows until it has its own scripting language and maybe its own text editor. Well, we have no plans to integrate Emacs into Nmap, but we decided that a scripting engine was essential. As Nmap grew, there were more and more neat ideas that we couldn't implement because adding them all would bloat the core of Nmap too much. Also, as Nmap became larger and more complex, it became harder for casual contributors to make improvements because they had to learn so many subsystems. And it was difficult to review all their patches because any mistake in the code could crash all of Nmap. With the scripting engine, people write to a relatively simple API and don't have to know anything about Nmap internals. Their code stays separate rather than bloating Nmap itself, and mistakes should only cause that particular script to fail.

We decided to use the Lua scripting language (https://www.lua.org/) because it is simple and small and easy to embed in a way that enables extremely fast network performance. It's mostly known in the gaming world, but some versions of other networking tools like Wireshark and Snort use it as well.

The scripting engine has been a great success! We now have more than 500 scripts, all documented at https://nmap.org/nsedoc/.

RF: From reading the Nmap 7.2 announcement, it sounds like there is an actual community involved with Nmap. Some open source project authors say "we" when really they are the only person writing code. Can you tell us about your code community?

FV: Yes, we are fortunate to have a community which helps in myriad ways. Not everyone is a programmer, but we also receive contributions in the form of people submitting unknown operating system or version fingerprinting results, helping to translate our documentation, creating art, or just answering questions on the mailing list. A good bug report is a gift, too.

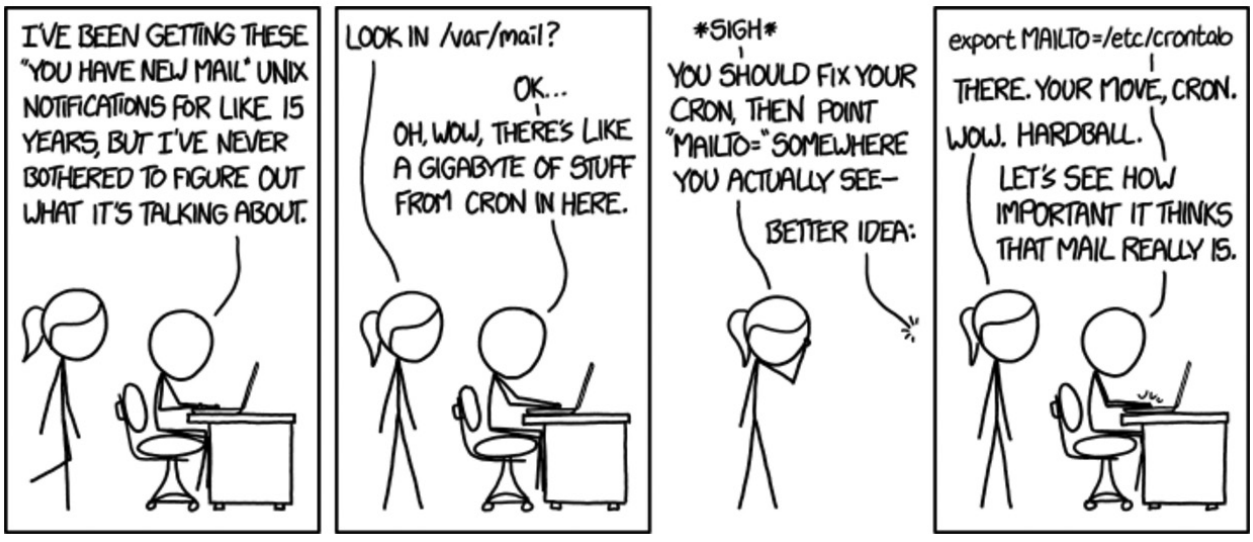
Contributing code has become harder as Nmap has grown more complex. NSE scripts, as discussed above, are a happy exception to that rule. We currently have more than 20 core code committers (https://svn.nmap.org/nmap/docs/committers.txt), although only a fraction are active at a given time. Every summer for the last 12 years Google has helped us fund many full-time interns (college or grad students) from all over the world. We've had 78 of them so far, and the program has been hugely suc-

cessful both in developing new features and in mentoring new contributors who sometimes help out for years afterward. Many of our former interns have actually become mentors for their own Nmap interns years later!

RF: I am guessing that you are an independent consultant and that the licensing of Nmap and pen-testing are how you support yourself. Is that true?

FV: That used to be the case, but Nmap grew so much and its development became so complex that I quit consulting and devoted myself to Nmap in 2002. I'm really happy with our dual-license program. It allows open source use of Nmap for free, but companies who want to distribute it with their commercial software have to pay a license fee. The fee also includes support, indemnification, and warranties that companies expect anyway. Not only does this fund the project, but it means that Nmap has many more users who may not realize it is even being used to perform network discovery under the covers of their proprietary software. We have roughly 100 licensees, from some of the largest companies in the world to tiny new startups. I'm always excited to see what innovative uses they can find for Nmap results. For example, exploitation frameworks can use Nmap to determine the remote OS so that they send the right shellcode for exploiting a vulnerability. They may only get one chance because the wrong payload could crash the service. Another neat usage is enterprise software installers that use Nmap to find all the systems compatible with their agent or management software.

XKCD



xkcd.com

Statement of Ownership, Management, and Circulation, 9/30/16

Title: *;login:* Pub. No. 0008-334. Frequency: Quarterly. Number of issues published annually: 4. Subscription price $90.

Office of publication: USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Headquarters of General Business Office of Publisher: Same. Publisher: Same.

Editor: Rik Farrow; Managing Editor: Michele Nelson, located at office of publication.

Owner: USENIX Association. Mailing address: As above.

Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: None.

The purpose, function, and nonprofit status of this organization and the exempt status for federal income tax purposes have not changed during the preceding 12 months.

Extent and Nature of Circulation			Average No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue (Fall 2016) Published Nearest to Filing Date
a. Total Number of Copies			2935	2800
b. Paid Circulation	(1)	Outside-County Mail Subscriptions	1278	1261
	(2)	In-County Subscriptions	0	0
	(3)	Other Non-USPS Paid Distribution	882	868
	(4)	Other Classes	0	0
c. Total Paid Distribution			2160	2129
d. Free Distribution By Mail	(1)	Outside-County	77	76
	(2)	In-County	0	0
	(3)	Other Classes Mailed Through the USPS	19	23
	(4)	Free Distribution Outside the Mail	512	288
e. Total Free Distribution			608	387
f. Total Distribution			2768	2516
g. Copies Not Distributed			167	284
h. Total			2935	2800
i. Percent Paid			78%	85%
Paid Electronic Copies			401	440
Total Paid Print Copies			2561	2569
Total Print Distribution			3169	2956
Percent Paid (Both Print and Electronic Copies)			81%	87%