FreeBSD Foundation April 2018 Update

Dedicated to supporting the FreeBSD Project and community

Message from the Executive Director

Dear FreeBSD Community Member,
We're excited to finally be surrounded by spring weather here at the Foundation. It's been a busy month including hosting our first ever FreeBSD Installfest/Bootcamp! Find out more about that, what our spring interns have been up to this term, the latest from Release Engineering, and more!

Enjoy!
Deb

April 2018 Development Projects Update

Another successful University of Waterloo co-op student term

At the end of April the Foundation's two co-operative education (co-op) students from the University of Waterloo, Mitchell and Arshan, will conclude their work terms and head back to school. The University of Waterloo is a pioneer in co-operative education, where students divide their time between University studies and practical work placements in industry. The Waterloo model divides the year into three four-month terms, with a total of eight study terms in Engineering and Computer Science. School and work terms run year-round, including over the summer. A typical four year degree thus takes an extra eight months, but upon graduation students have a total of two years of real work experience.

Last month I wrote about one of Mitchell's main projects, updating the FreeBSD port of the Syzkaller kernel system call fuzzing tool. Mitchell's working on a blog post that describes his experience as a newcomer to FreeBSD development working for the FreeBSD Foundation, so stay tuned for that. This month I'll focus on one of Arshan's projects. Arshan has worked on a few projects over the term, including supporting the Foundation's effort on the recent Spectre &

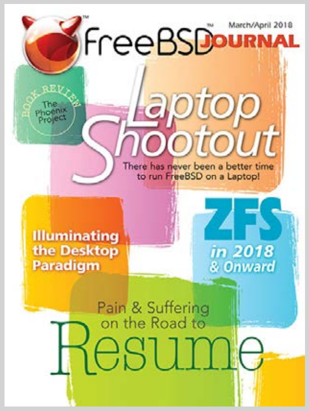
Upcoming Events

Rootconf 2018
May 10-11, 2018
Bangalore, India

BSDCan 2018 FreeBSD Developers Summit
June 6-7 , 2018
Ottawa, Ontario, Canada

BSDCan 2018
June 6-9, 2018
Ottawa, Ontario, Canada

FreeBSD Journal



The March/April issue of the *FreeBSD Journal* is now available. Don't miss articles on Illuminating the Desktop Paradigm, ZFS 2018 and Onward, and more!

New Feature! Browser-Based subscribers now have the ability to download and share PDFs of the articles!

Sample Issue! If you've ever wanted to read through an entire issue of the FreeBSD Journal, now's your chance. Download the sample issue and be sure to share with your friends and colleagues.

Not a subscriber? Sign up today!

See what others are saying about the Journal:

"Awesome! This is the best way to popularize FreeBSD!!" San Jose, California

"I've found it really practical, and great reading...it caters to all levels of users." Brooklyn, NY

Why Choose FreeBSD?

"At NeoSmart Technologies, we don't build just applications on top of the operating system – we build applications for the OS itself. Our Easy Recovery Essentials® line of bootable system repair CDs is used by thousands of end users, IT professionals, and system administrators around the world on a daily basis to detect and correct where possible issues with the users' hardware, boot configuration, operating system, and applications in a safe, stable, and reliable manner.

Easy Recovery Essentials was purposely written in a mix of (mostly) cross-platform C/C++, a dusting of assembly, and an html user interface in order to allow us to evaluate different "host" platforms to boot and run our software on hundreds of thousands of different PCs — decades old and breaking-edge new — from all the various manufactures and OEMs. Unlike deploying an operating system on servers in a datacenter or in the cloud where the hardware is fully under your control, we actually needed something that would reliably run, GUI and all, on end-users' hardware with minimal issues… a tall order.

After trialling both Windows® PE and various Linux distributions in production for several years, we found that only FreeBSD offered a rock-solid, open, well-architectured, and cohesive platform at the lowest level upon which we could reliably base our software and

Meltdown mitigation work by benchmarking and testing in-progress patches, and a number of other small projects and tasks. I asked him about the project he found most interesting this term.

Ed: What project really caught your attention this term?

Arshan: A week after PI day, in the staff meeting, I heard you talk about the Raspberry PI 3B+ that was newly released. You mentioned that FreeBSD doesn't support the network chip used in the new RPi, and that we need someone to write a driver for it. That caught my attention as I was looking for some driver project to work on.

Ed: What was it that attracted you to this project?

Arshan: I was always curious to learn about the low-level details of how hardware interacts with software. Because of this, device drivers were attractive to me in general. With this particular project, I think the fact that it was related to RPi made it even more interesting for me. RPi is a popular embedded platform, and a contribution towards RPi would have a big impact on the FreeBSD community. To be honest I didn't think I would be able to get it to a working state during the remaining days of my coop, but thankfully and luckily, it did and I'm grateful for that.

Ed: What was the most challenging part of the driver porting effort?

Arshan: The whole process was pretty linear, I didn't get stuck in anything for a long time that would make it tedious. I think it was mostly because FreeBSD has a pretty awesome USB and a USB to Ethernet stack that did some of the heavy lifting for me. The existing drivers on the source tree were also well documented. Reading those definitely helped a lot.

Ed: Describe your interaction with the FreeBSD community as you worked on the driver port.

Arshan: The community was really helpful. I had to contact the developer of the driver for the USB-Ethernet controller on older RPi's on FreeBSD. Although he is no longer a committer of the project, he was still really timely on his responses. He was more than happy to help me with some of the details of the driver. People at Microchip, and the Raspberry Pi Foundation were also really helpful.

Ed: What's the current state of the driver? What's next?

Arshan: The patch is currently in review; with the patch users should be able to connect to the internet with their Raspberry Pi's using an ethernet cable with no problem. However, the driver is at a pretty minimal state right now. There are several features that the chip supports such as RX/TX checksumming, VLAN tag removal, and a bunch more that are not currently enabled on the driver. I'm planning on adding support for all these features in the near future. I'm currently working on RX/TX checksumming.

expect it to “just work” for almost all of our customers, everywhere, all the time. The logical and centralized-yet-modular codebase makes it easy to understand how the operating system works and to extend it with the functionality we need without breaking things, allowing us to focus on developing our product rather than resolving issues with the bootable host environment.

Last but not least, FreeBSD’s completely free, no-strings-attached licensing model let us incorporate and take advantage of FreeBSD as we saw fit without needing a team of lawyers to tell us whether or not we could use FreeBSD to do amazing things – and made it easy for us to simply contribute virtually all of our improvements, ports, integrations, and bugfixes right back to the community under the same, open license we obtained them under in the first place.

For us, sticking with and continuing to support FreeBSD is a no-brainer. We are proud to sponsor FreeBSD every year, and readily recommend it “hands down” when asked for our recommendation for a fast, battle-tested, and reliable platform."

– Mahmoud Al-Qudsi, [NeoSmart Technologies](#).

2 Issues for the Price of 1!
Order a Risk-Free Trial Subscription
ADMIN
Network & Security
.NET on Linux
shop.linuxnewmedia.com

Ed: Overall, how did you find the term? Any final thoughts?

Arshan: This has been my fourth work term, and the best one so far. I wish I had more time with the Foundation as I think I really got up to speed in the second half of my work term. I really liked the freedom I had here. There are so many interesting projects to work on, and lots of great resources and people to reach out to for help. I think and I hope that I remain actively involved with the FreeBSD project in the future.

-- contributed by Ed Maste

Fundraising Update: Thank You

On behalf of the Foundation, I want to extend a heartfelt thank you to everyone who has made a donation to us this year. So far, we’ve raised $170,000 from 252 donors!

I’m also pleased to welcome our 2018 Partners including our Iridium Partner [NetApp](#) and Silver Partners: Microsoft, NeoSmart Technologies, and Tarsnap. Companies like these receive exclusive benefits, while helping to fund our efforts. We rely on these partnerships to continue supporting key areas such as:

- Accelerating operating system improvements and quickly stepping in to fix critical issues.
- Improving developer tools, testing, and automation.
- Managing the release engineering efforts to provide reliable and timely releases.
- Leading the security team efforts.
- Providing outreach and advocacy around the world, to continue increasing the number of FreeBSD users and contributors.

You can find out more information about the program [here](#).

If your company is using FreeBSD, then you are benefiting from the work we are doing. Please help us continue this work by becoming a FreeBSD Foundation Partner today!

We depend not only on corporate donors, but on individual donors too! In fact, most of our donors are individuals who want to show their support for the community, as well as, the work we are doing. If you haven’t made a donation this year, please consider [making](#) one now!

Thank You!

-- contributed by Deb Goodkin

FreeBSD Foundation Installfest/Bootcamp

101.0

I couldn't be more thrilled for the outcome of our first ever Foundation-hosted FreeBSD Bootcamp on April 16. I've talked for years about developing workshops and training material for introducing new people to FreeBSD, and by working with some local FreeBSD contributors, we finally made it happen!

It definitely took some focus, evenings, and weekends to put it all together, but now we'll have this first introductory workshop available for people to hold their own installfests and bootcamps around the world.

Two weeks ago, we had a planning meeting at the Foundation office, to discuss what we should cover in a 2-3 hour workshop. The goal was to come up with enough relevant content to engage people and interest them in wanting to use or contribute to FreeBSD.

We continued working on the curriculum as we played around with our own FreeBSD setups, adding what we thought would be interesting to our working document.

After multiple people reviewed and provided feedback, we finalized our material. The process was slightly less daunting, knowing this was going to be a learning opportunity for us to improve on the material based on the experiences and input we received from the participants.

I invited 6 female engineering friends, who were not only interested in learning about FreeBSD, but also open to being our guinea pigs for this trial run of the workshop.

We also designed a fun t-shirt for this inaugural event, as you can see below.

We decided to have everyone install FreeBSD on VirtualBox to allow us to speed through the installation process, and cover more material.

After introducing ourselves and eating some pizza, we began by giving a short "Introduction to FreeBSD" presentation. We then downloaded

FreeBSD, which led us to the first lesson we learned, that it took a long time! Next time, we will have people start the download first, then we can give the FreeBSD talk in parallel.

Once the download was complete, we installed FreeBSD, went through the initial setup, rebooted, and were officially running FreeBSD! That's when the fun really began. Why? Most of the attendees started out their careers using UNIX, and one participant talked about feeling young again, as she navigated the command line. She also spoke about how this exercise highlighted that when we mainly use MacOS and Windows, we're removed from understanding the foundation of computers and the hardware involved. Most people nowadays want to get their smartphones, tablets, and computers running with minimal intervention. That's how I am most of the time, just wanting things to work, and getting frustrated when they don't. But, understanding the foundation of whatever I'm using, whether it's a computer, bicycle, or software, makes it so much easier to fix or debug when you understand how those things work.

Once we were running FreeBSD, we installed xorg, lumina, firefox, sudo, libreoffice, and plexmediaserver packages, added some configuration information, and ran lumina. We completed the workshop by accessing the internet.

In the end, we realized we had more material to cover than time allowed, which is perfect for kicking off the FreeBSD 202.0 Workshop!

Again, I was pleased with the turnout for this event and am now updating the materials, so we can make this workshop available to anyone who would like to host one. I'm also thrilled that this group of participants is already planning for our next meeting where we can continue learning about FreeBSD and hopefully start contributing to the Project!

- contributed by Deb Goodkin

Guest Blog Post: A Look at SDN Network Emulator, Mininet

At this year's AsiaBSDCon, I presented a talk about a SDN network emulator called Mininet, and my ongoing work to make it more portable. That presentation was focused on the OpenBSD version of the port, and I breezed past the detail that I also had a version or Mininet working on FreeBSD. Because I was given the opportunity, I'd like to share a bit about the FreeBSD version of Mininet. It will not only be about what Mininet is and why it might be interesting, but also a recounting of my experience as a user making a first-time attempt at porting an application to FreeBSD.

First, a bit of context --

Thanks to its status as a buzzword, SDN, or software-defined networking, has different meanings to different people. The interpretation that I follow is that SDN is an approach to network design where its traffic-forwarding elements (datapaths or switches) expose a control channel through which a network control application with a global view of the network can steer it as if it were one entity. With this structure, managing the network means interacting with this logically central application, rather than dealing with each individual device and hoping that the network's behavior converges to what you want.

Mininet started off as a tool used by academic researchers to emulate OpenFlow networks when they didn't have convenient access to actual networks. Because of its history, Mininet became associated strongly with networks that use OpenFlow for their control channels. But, it has also become fairly popular among developers working in, and among several universities for research and teaching about, SDN.

I began using Mininet as an intern at my university's network research lab. I was using FreeBSD by that time, and wasn't too happy to learn that Mininet wouldn't work on anything but Linux. I gradually got tired of having to run a Linux VM just to use Mininet, and one day it clicked in my mind that I can actually try porting it to FreeBSD.

From a user's perspective, Mininet creates a SDN-capable network made of nodes and links. There are several types of nodes, including switches, the controller(s) that the switches are managed by, and end hosts. A developer can, for example, test their SDN application by replacing the controller with their own and running scripts on their hosts to send and receive various types of traffic, seeing how it copes.

Mininet creates a topology using the resource virtualization features that Linux has. Specifically, nodes are bash processes running in network namespaces, and the nodes are interconnected using veth virtual Ethernet links. Switches and controllers are just nodes whose shells have run the right commands to configure a software switch or start a controller application. Mininet can therefore be viewed as a series of Python libraries that run the system commands necessary to create network namespaces and veth interfaces, assemble a specified topology, and coordinate how user commands aimed at nodes (since they are just shells) are run.

Coming back to the port, I chose to use vnet jails to replace the network namespaces, and epair(4) links to replace the veth links. For the SDN functionality, I needed at least one switch and controller that can be run on FreeBSD. I chose OpenvSwitch(OVS) for the switch, since it was available in ports and is well-known by the SDN world, and the Ryu for the controller since it's being actively developed and used and supports more recent versions of OpenFlow. (Mininet does give a user multiple choices of switches and controllers, but aside from OVS, none of them seemed to have been ported or well-tested outside of Linux). At the end of this exercise, I had what the creator of Mininet, Bob Lantz, would call "version 1", which can be used to run customized networks and for development, but doesn't support some of the fancier features like the drag-and-drop topology creator, or traffic-shaped links.

I have also been in touch with Bob, and have discussed the possibility of upstreaming my work. Although he was excited about it, I was asked about a script for creating VMs with Mininet preinstalled, and continuous integration support for my fork of the repository. I started taking a look at the release scripts for creating a VM, and after seeing that it would be much easier to use the scripts if I can get Mininet and Ryu added to the ports tree, I also tried a hand at submitting some ports. For CI support, Mininet uses Travis, which unfortunately doesn't support FreeBSD. For this, I plan to look at a minimalistic CI tool called contbuild, which looks simple enough to get running and is written portably.

This is very much a work-in-progress, and one going at a glacial pace. Even though the company that I work for does use Mininet, but doesn't use FreeBSD, so this is something that I've been working on in my free time. Earlier on, it was the learning curve that made progress slow. When I started, I hadn't done anything more than run FreeBSD on a laptop, and uneventfully build a few applications from the ports tree. Right off the bat, using vnet jails meant learning how to build and run a custom kernel. This was the easy part, as the handbook was clear about how to do this. When I moved from using FreeBSD 10.3 to 11, I found that I can panic my machine by quickly creating and destroying OVS switches and jails. I submitted a bug report, but decided to go one step further and actually try to debug the panic for myself. With the help of a few people well-versed in systems programming and the developer's handbook, I was able to come up with a fix, and get it accepted. This pretty much brings my porting experiment to the present day, where I'm slowly working out the pieces that I mentioned earlier.

In the beginning, I thought that this Mininet port would be a weekend project where I come out knowing thing or two about using vnet jails and with one less VM to run. Instead, it became a crash course in building and debugging kernels and submitting bug reports, patches, and ports. It'd like to mention that I wouldn't have gotten far at all if it weren't for the helpful folks, the documentation, and how debuggable FreeBSD is. I enjoy good challenges and learning experiences, and this has definitely been both.

-- contributed by Ayaka Koshibe

April 2018 Release Engineering Update

The FreeBSD Release Engineering team started the 11.2-RELEASE cycle on April 20, beginning with the code slush. The code slush is the point at which commits to the relevant stable branch do not require explicit approval from the Release Engineers. However, it is requested that new features and major changes be avoided while we transition to the code freeze part of the cycle, limiting the number of intrusive changes to the tree before changes require approval.

FreeBSD 11.2-RELEASE will be the third release from the stable/11 branch and builds on the reliability and stability of 11.1-RELEASE. FreeBSD 11.2 is scheduled to be released late June, provided there are no major issues that warrant extending the schedule. The complete schedule is available on the FreeBSD Project website.

- contributed by Glen Barber

Submit Your Work! Check out open CFPs

Presenting at a conference is an excellent way to spread the word about the work you're doing, while raising awareness for FreeBSD. Below is list of upcoming Calls For Proposals

Open Source Summit North America
August 29-31, 2018
Vancouver, British Columbia, Canada
The Call for Proposals for the 2018 Open Source Summit North America is now open. More information and a list of suggested topics can be found here.
Submission Deadline: April 29, 2018

USENIX LISA18
October 29-31, 2018
Nashville, TN
The LISA18 Program Committee is looking for industry leaders and people on the front lines to propose topics that demonstrate the present and future of operations. LISA submissions should inspire and motivate attendees toward action that improves their day-to-day work as well as the tech industry as a whole. Find out more here.
Submission Deadline: May 24, 2018

EuroBSDcon 2018
20-23 Sept 2018
University Politehnica of Bucharest, Romania
The EuroBSDcon program committee is inviting BSD developers and users to submit innovative and original talk proposals not previously presented at other European conferences. Find out more here.
Submission Deadline: June 17. 2018

Open Source Summit Europe 2018
October 22-24, 2018
Edinburgh, Scotland, UK
The Call for Proposals for the 2018 Open Source Summit Europe is now open. More information and a list of suggested topics can be found here.
Submission Deadline: July 1, 2018

-- contributed by Anne Dickison