

A portrait of Barbora Buhnová, a woman with blonde hair tied back, wearing a white headband with small white flowers and a white button-down shirt with a small blue polka-dot pattern. She is smiling and looking towards the camera.

# RH RQ

Bringing great research ideas  
into open source communities

## Barbora Buhnová

*On founding Czechitas and  
opening the doors of tech*



Constant-time cryptography

The elastic bare metal cloud

Linux-based unikernels

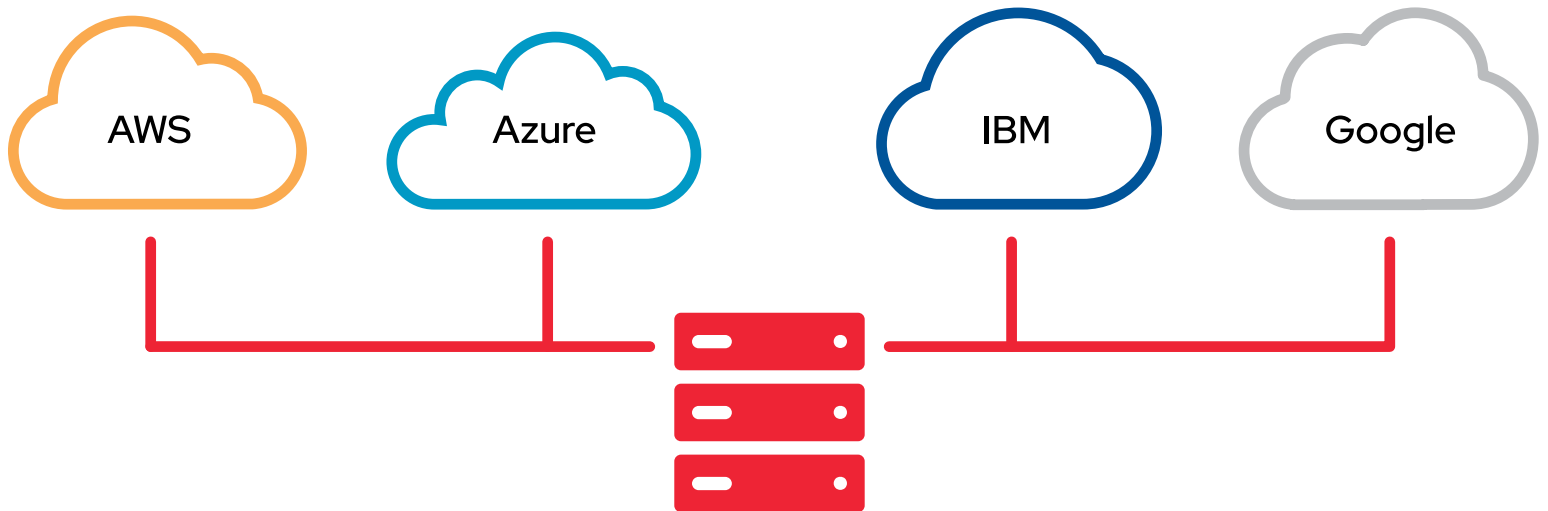
Machine learning  
accessibility &



Red Hat  
Research Quarterly

Volume 3:3 | November 2021 | ISSN 2691-5278

# Clouds that compete



should still connect.

| Our code is open\_

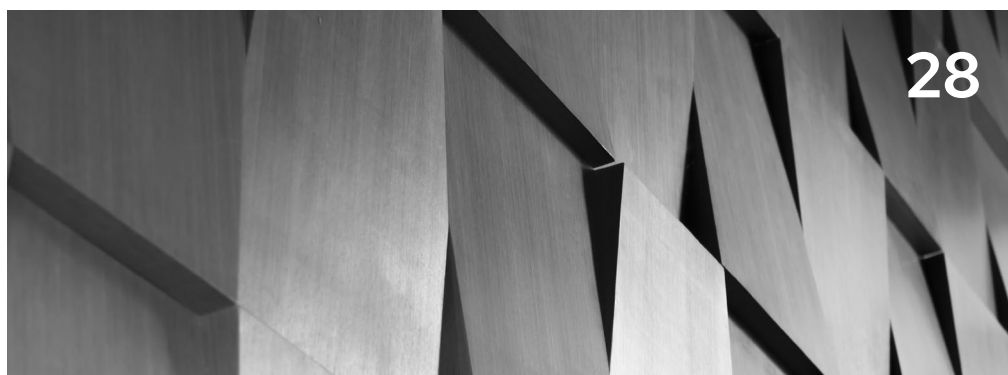
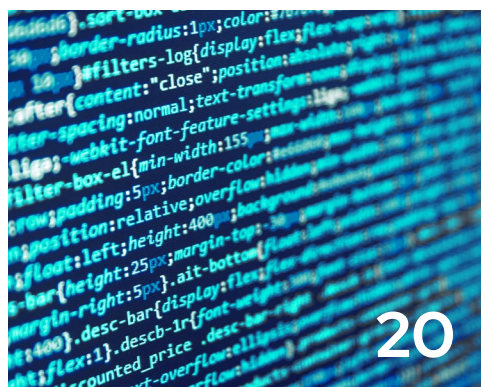


**Red Hat**

[redhat.com/ourcode](https://redhat.com/ourcode)



## Table of Contents



**ABOUT RED HAT** Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to provide reliable and high-performing cloud, Linux®, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

NORTH AMERICA  
1 888 REDHAT1

EUROPE, MIDDLE EAST,  
AND AFRICA  
00800 7334 2835  
europe@redhat.com

ASIA PACIFIC  
+65 6490 4200  
apac@redhat.com

LATIN AMERICA  
+54 11 4329 7300  
info-latam@redhat.com



facebook.com/redhatinc  
@redhatnews  
linkedin.com/company/red-hat

### Departments

- 04** From the director
- 06** News: Red Hat-BU Collaboratory awards
- 07** News: Test case prioritization
- 09** News: Progress in programmable networks
- 35** How COVID made our world smaller
- 38** Research project updates

### Features

- 12** Opening the doors of tech: an interview with Barbora Buhnová
- 20** The need for constant time cryptography
- 25** The elastic bare metal cloud is here
- 28** Creating a Linux-based unikernel
- 32** Making machine learning accessible across disciplines

From the Director

## Let's help more programmers get into the groove

by Hugh Brock



### About the Author

**Hugh Brock** is the Research Director for Red Hat, coordinating Red Hat research and collaboration with universities, governments, and industry worldwide. A Red Hatter since 2002, Hugh brings intimate knowledge of the complex relationship between upstream projects and shippable products to the task of finding research to bring into the open source world.


**A**s a percussionist, I find myself thinking probably more than most people about time—although what a percussionist means by “time” is more like “meter” or “rhythm” in common parlance. A drummer has “good time” if they can maintain a regular, predictable pulse—a “groove”—while still doing a whole bunch of unrelated things. It’s a skill, and an art too, because of course time is also a consensus among band members, one that the poor drummer must constantly manage and reinforce without being a jerk about it. Once you learn to listen for time—for regularity that is reliable and well communicated, but also flexible when needed—you will have a whole new appreciation for what the drummer is doing.

This notion of time is what struck me as so interesting about this issue’s feature on constant-time cryptography. It turns out that a crypto implementation whose execution time varies depending on what you feed to it is inherently leaky. By looking at the inputs to a non-constant-time crypto function, an attacker can infer enough about the secret key the function depends on to guess the key, often trivially. Like a drummer who gets distracted by a solo and rushes or drags the time, the crypto function reflects back information about the secret it is protecting. Unfortunately for drummers, it is a lot easier to write a constant-time crypto function

than it is to learn truly great time—but no less important to the success of the programmer.

The question of what makes a successful programmer also emerges in this issue’s interview with Barbora Buhnová. Buhnová is a leader at the Faculty of Informatics at Masaryk University in Brno, Czech Republic, and one of the founders of Czechitas, a nonprofit organization aimed at making IT skills more accessible to women of any age. She believes the key to bringing more girls and women into tech is, first, about simply seeing value in different mindsets and problem-solving strategies and, second, about creating a critical mass that changes expectations. We need to go from “Nobody like me does this” to “I see lots of people like me doing this.”

The same principle applies to AI and machine learning, which is becoming not just an area of interest but a requirement across multiple disciplines. In his piece “Making machine learning accessible across disciplines,” Red Hat engineer Marek Grác talks about designing the Applied Machine Learning course at Masaryk University. “We wanted to give students an opportunity to learn new things without getting into a routine of simply opening an editor and coding something, so we had students creating annotation guidelines, fixing bugs in existing source code, trying to create better rules for simple games, creating chatbots, or trying to figure out which parameters might influence whether a person will get a mortgage,” he says.

In other words, learn to do what you need to do without having to turn yourself into a full-time programmer. As the use of programming and AI becomes more and more common across disciplines, this approach is going to become the more common one, I think. Courses like Marek’s represent a great start. 





# THE UNIVERSAL AI SYSTEM FOR HIGHER EDUCATION AND RESEARCH

## NVIDIA DGX A100

Higher education and research institutions are the pioneers of innovation, entrusted to train future academics, faculty, and researchers on emerging technologies like AI, data analytics, scientific simulation, and visualization. These technologies require powerful compute infrastructure, enabling the fastest time to scientific exploration and insights. NVIDIA® DGX™ A100 unifies all workloads with top performance, simplifies infrastructure deployment, delivers cost savings, and equips the next generation with a powerful, state-of-the-art GPU infrastructure.

Learn More About **DGX** @ [nvda.ws/dgx-pod](https://nvda.ws/dgx-pod)

Learn More About **DGX on OpenShift** @ [nvda.ws/dgx-openshift](https://nvda.ws/dgx-openshift)

News



About the Author

**Shaun Strohmmer**

is the editor of the *Red Hat Research Quarterly*. She has worked as a writer and editor in academic publishing for over twenty years, and since 2014 she has focused on software development, cybersecurity, and computer science.

## Red Hat Collaboratory at Boston University granting major awards

The Collaboratory solicited proposals from BU faculty for both large and small research projects to drive innovation for the open hybrid cloud.

by Shaun Strohmmer

**T**he Red Hat Collaboratory at Boston University has moved into a new phase of identifying and funding promising research projects, and the selection process is currently underway. The submission deadline was October 1, 2021, and final award announcements will be made December 15, 2021.

The purpose of the Collaboratory is to connect BU faculty and students with industry engineers working in open source communities. These awards will provide both substantial financial support and the support of Red Hat engineers to the winning proposals.


Projects are expected to focus on problems of distributed, operating, security, or network systems whose solution shows promise for advancing their field and impacting industry. Software developed in the course of this research must be made available under an open source license, and all results will be made publicly available.

The Collaboratory will fund projects at three levels: large (< \$500K per year), involving multiple faculty and students; small (< \$150K); and speculative (< \$75K). Large-scale projects are expected to be highly visible, and will be affiliated with a

specific Red Hat business unit, which will provide additional resources. Small-scale projects will focus on an individual researcher's area of expertise and will include collaboration with at least one Red Hat engineer. Speculative projects may include fundamental systems research, work relevant to the Open Research Cloud Initiative, and high-risk projects. Researchers of speculative projects do not need to have the committed involvement of a Red Hat engineer, but initiating university-industry collaboration through these projects remains a priority.

Projects have a timeline of one to two years, depending on scale, and are renewable annually after the award period ends. The Collaboratory's goal is to encourage multi-year projects while making sure that they continue to progress and providing support to projects that are running into problems.

BU researchers and Red Hat technical experts will review applications, making selections based on feasibility, potential visibility, potential impact, novelty, contribution to diversity in Collaboratory research, and relevance to the Collaboratory and its infrastructure, all within the context of the proposed budget.

As part of its planned growth, the Collaboratory looks to fund two large-scale projects, six small-scale projects, and six speculative projects annually. 

## Test case prioritization: towards high-reliability continuous integration

A new project seeks to make effective testing more compatible with expeditious releases.

by Ilya Kolchinsky

In August 2021, a team of graduate students from [IDC Herzliya](#), a leading research college in Israel, began working on a test case prioritization (TCP) project under the guidance of senior engineers from Red Hat.

The goal of the TCP project is to create a novel tool based on machine learning (ML) that solves the test case prioritization problem in software regression testing. Automatic regression testing is a crucial step of any CI/CD pipeline. The primary purpose of this testing is to detect bugs and defects introduced by recent changes as early as possible while keeping verification costs low. The ability to perform regression testing efficiently and effectively (i.e., within a small time frame, yet catching the majority of bugs) would allow developers to rapidly deliver reliable software updates to users.

Unfortunately, the regression testing process in modern large-scale software products tends to be too complex and

cumbersome to meet this objective. As the size of software increases, the test suite also grows bigger and requires more time and resources to be fully executed. In many cases, the time to run the entire test suite can reach three or four days or even a week. Consequently, executing all available tests during the CI/CD regression testing procedure is highly impractical and, in many cases, completely infeasible.

To address this issue, TCP methods have emerged. TCP aims to order a given set of test cases such that the earlier a test appears in the resulting order, the higher the probability that this test will detect a bug or a fault introduced by the given code change. Provided such an ordering exists for the entire test suite, the regression testing procedure can iterate over it starting from the beginning and advancing until a predefined limit on the maximal testing time (say, one minute) is reached. Because the most significant test cases are executed first, the chances of early






## About the Author

**Ilya Kolchinsky** is a research scientist with Red Hat Research and Technion – Israel Institute of Technology. He has a PhD and BSc in computer science, both from the Technion. Ilya's research interests span a wide range of topics in big data processing, such as distributed event-based systems, data stream mining, and applications of AI and machine learning in stream processing engines.

fault detection rise even when executing only a negligible part (say, 1%) of the entire test suite.

In recent years, TCP solutions have been adopted by major industry players and have spawned a wave of academic research projects. However, the full potential of TCP in regression testing is yet to be explored. As of now, the most widely used approaches are mainly based on heuristic search strategies and/or code coverage methods. In contrast, methods based on machine learning in general and deep learning in particular are barely explored. Closing this gap and devising a ML-based TCP tool is the primary goal of this project.

The project team is now progressing towards the first major milestone: implementing an open source proof-of-concept prototype that replicates the state-of-the-art academic results in this area. This milestone is expected to be reached by November 2021. As an immediate next step, the team will be looking to improve the state of the art. To that end, a variety of machine learning and data mining methods will be considered. 



*Those interested in finding out more about the TCP project and/or looking for collaboration opportunities are kindly invited to contact Dr. Ilya Kolchinsky ([ikolchin@redhat.com](mailto:ikolchin@redhat.com)) or Gil Klein ([gklein@redhat.com](mailto:gklein@redhat.com)).*



TCP aims to order a given set of test cases such that the earlier a test appears in the resulting order, the higher the probability that this test will detect a bug or a fault introduced by the given code change.

## News



# Programmable networking project reports on its first year of progress

by Toke Høiland-Jørgensen

## About the Author

**Toke Høiland-Jørgensen**

is a Principal Kernel Engineer at Red Hat working on networking and BPF. He holds a PhD on the topic of network performance and bufferbloat.

Researchers from Red Hat and Karlstad University, Sweden, have recently finished their first year of work on enhancing the performance of the eXpress Data Path (XDP), a data path integrated into the Linux kernel that permits flexible programmable networking. The group's year one report, "Building the next generation of programmable networking—powered by Linux," was released September 10, 2021.

## TWO PROBLEMS, TWO GOALS

Given the potentially wide scope for the project, researchers prioritized two areas: enhancing XDP with queuing capabilities and using BPF/XDP for efficient latency monitoring.

While XDP excels in forwarding packets, it has no mechanism for queuing or reordering of packets and cannot implement traffic scheduling policies. To remedy this, the researchers are experimenting with ways of adding programmable packet scheduling to XDP. The goal is to allow programmers to define their packet schedulers using BPF while benefiting from the XDP fast data path.

Currently the team is prototyping the Linux BPF queueing implementation and implementing a prototyping framework to execute different scheduling algorithms quickly and try different API constructs. Using the framework, they will continue to experiment with different ways of constructing packet scheduling algorithms and improve

their ease of use. The report illustrates the new packet design scheduling framework, using BPF hooks for enqueue, dequeue, and timer hooks.

The researchers are also exploring ways to employ BPF/XDP for latency monitoring. If successful, BPF could alleviate several problems involved in measuring latency using tools like ping: additional network overhead, lack of scalability, and a limited view of possible latency causes. Because using


## ABOUT THE PROJECT


**The XDP project is a collaborative effort with participants from multiple organizations**

The core project team includes:

- **Red Hat engineers**  
Toke Høiland-Jørgensen and  
Jesper Dangaard Brouer
- **Karlstad University professors**  
Anna Brunstrom and Per Hurtig
- **PhD students**  
Freysteinn Alfredsson and Simon Sundberg

Staff members from Red Hat and Karlstad, as well as representatives from Ericsson, have also participated in some meetings. Researchers plan to make results available as open access, open source software and open data.

 *Professor Brunstrom was interviewed for the August 2021 issue of RHRQ; see "The right idea at the right time: networking researchers use open source for real-world results."*

passive ping (pping) can mitigate some of those issues, researchers developed an implementation of pping using BPF (e-pping). Initial results detailed in the report suggest that e-pping introduces negligible CPU overhead. 



Those interested in reviewing the project and providing comments are encouraged to do so, either by going to the [GitHub repository](#) or by emailing [toke@redhat.com](mailto:toke@redhat.com).

# RH RQ

*Read current articles*

*Browse back issues*

*Subscribe to print or digital*

[research.redhat.com/quarterly](https://research.redhat.com/quarterly)



# Red Hat Research Quarterly





# DEVCONF.cz

open source community conference

virtual  
event

January 28-29, 2022

📍 Brno and all around the world

Analysis, Testing and Automation

Cloud & Hyperscale • Open Source Education

Linux distribution • HPC, Big Data and Data Science

Modern Software Development • Edge Computing

Future Tech and Open Research

Open Source UX/Design

# SAVE THE DATE

[www.devconf.info/cz](http://www.devconf.info/cz)





# Opening the doors of tech

*Why diversity is critical  
to the future of computing*

*Interview by Matej Hrušovský*



## Interview

**R**ed Hat Research University Program Manager **Matej Hrušovský** interviewed **Barbora Buhnová**, Associate Professor and Vice Dean for industrial partners at Masaryk University, Faculty of Informatics in Brno, Czech Republic. She is also the chair of the Association of Industrial Partners of Masaryk University, Faculty of Informatics, and is a co-founding and governing board member of Czechitas, a nonprofit organisation aimed at making IT skills more accessible to youth and to women of any age.

**Matej Hrušovský:** When did you start getting interested in the issue of women's participation in computer science, and when did you start taking action?

**Barbora Buhnová:** Founding Czechitas, the nonprofit organization we are running in the Czech Republic, changed my perspective on women in tech in a big way. Many believe that girls are not really interested in computing, and that's why there are so few women represented in computing. But with Czechitas, we see a different picture. When we introduced coding classes for girls, they became so successful that within one day of opening a new course for 30 women, we could get 350 registered.

It was a complete mind shift for many of us, recognizing that girls are extremely interested in learning coding and getting into the IT world. They were just waiting for a different kind of invitation, which we were able to offer.

**Matej Hrušovský:** What was the spark for the idea to start Czechitas in the first place?

**Barbora Buhnová:** We have heard many girls say that they find coding interesting, but they believe that it is too late to start. That is a theme we've seen quite a lot: "It's too late for me. I was never encouraged to try it when I was younger, and now I am twenty and it's too late." So that was the spark—let's give it a try. Let's just spend one day with hands-on

coding, and you can find out for yourself if you have this passion and want to work on it.

Our first course was on Ruby On Rails, within the international Rails Girls initiative. But after that course, many participants were really excited about continuing. So the people who were engaged in organizing the course and teaching the course decided to create the continuation.

We did some introductory coding, some web development, things like mobile development. We did Android development very early—maybe too early because for somebody who has no coding experience, starting with Android development was very ambitious—but it was a lot of fun. There's a spirit of exploring something that might be too difficult, but being in it together.

**Matej Hrušovský:** Where do you see Czechitas going in the future?

**Barbora Buhnová:** We have different pillars in which we are trying to achieve certain goals. One of those is popularization: changing the belief that IT is only for a very limited group of highly intelligent people. We steer away so many talented people who choose different disciplines because they believe that IT is too difficult for them.

The second direction is transitioning into tech: changing a career or changing a study direction. For example, a young woman studying economics could change her direction to tech, because she



### About the Author

#### **Matej Hrušovský**

has been with Red Hat for more than eight years, six of which have been spent managing the university program in EMEA. Aside from attracting new talent mainly from universities and schools, the core of Matej's job is to find and put the right people from Red Hat and academia in the same room together.



already has a skillset we need. There is no reason for her not to choose it, except her own beliefs about the industry and her chances to succeed.

This also applies in terms of career change. In the Czech Republic, we are known for having very long maternity leave. The average is six years, since many women have a second child while they are on leave with the first. That is six years of staying home with zero connection to a career. What we see is that these women often reconsider their career direction. They are thinking about using this time to learn something new and then starting a brand new career. That is a huge opportunity, but at the same time it is very difficult for them.

**Matej Hrušovský:** Why do you think there is this barrier for women in society at all?

**Barbora Buhnová:** It is a combination of factors, and it is different in different countries. I am now Vice Chair of the European Network for Gender Balance in Informatics (EUGAIN), which represents thirty-eight European countries. One of the things we do is try to understand these differences among countries. For instance, how do these issues differ based on the structure of the society, such as countries in which multiple generations of a family live together.

In the Czech Republic, we ran a study with 150 participants trying to compare the directions and perspectives of two personas: a woman who is in tech,



and a woman who wants to learn tech now that she is an adult. For the second persona, we were asking, why isn't she in tech in the first place?

We identified a leaky pipeline here in the Czech Republic. It starts with a lack of encouragement when they are little girls. That's something the participants all referred to. They say, "It never crossed my mind because I've never been encouraged to even think about it." Access to encouragement is a very subtle change that may make a huge difference.

The second interesting factor is that many girls are multidisciplinary. They like languages, and they like history, and they like biology, and they like computing. They like math and they like arts. Given that wide range of interests, if you get a really bad teacher in computing,

you just forget about that direction and focus on the other directions you find interesting. A really good teacher is a critical influence. That can be a challenge when some teachers are not excited about technology themselves.

Families can also be an influence. Some girls we spoke to were interested in computing, but their family was discouraging them and directing them towards a career where it was easier for the family to imagine they would succeed. Their mom might say, "Maybe you would be happier as a nurse," in an attempt to protect her from going into this unknown world where she cannot imagine a successful, happy place for her little girl.

So it's not about bad intentions; it's just the stereotypes parents



"In the end we want to become more open, because we need more people in tech. We need boys, we need girls, we need everybody who is interested..." –Barbora Buhnová

have of computing as a world where their girls might not thrive. For that reason, it's crucial for Czechitas to work on popularization with adult women, because they will be those mothers and role models for their little daughters. The way they talk to their daughters about computing is crucial.

These elements build on each other. When a girl gets over these hurdles and finds herself at high school or university studying computer science, she wants a sense of belonging. In the Czech Republic, we know that the ratio of girls to boys in computing is very low, so the opportunity to identify with others who are successful in that field is hard to find. And many girls in that stage then start mimicking the majority group and forgetting about their own talents and strengths.

**Matej Hrušovský:** It's a fear of standing out, right?

**Barbora Buhnová:** Yes. And in that case, they're also trying to be good at what they see the boys are good at. They diminish their own potential by trying to mimic someone else. That hurts their confidence, because they will never be able to mimic someone else's strengths as well as they can use their own. Different people have different perspectives and will see what another person misses. Girls need to understand that it is valuable to have their unique talents that they can bring to the table and really stand out. That is the true power of diversity.

I'm now talking about myself a bit. For a big part of my early career, I was really trying to be a great specialist.

But I will never beat specialists at being a specialist when I'm not a specialist myself. I'm a generalist. I'm a multidisciplinary mindset person. I need to see the big picture. And of course I used to struggle with my confidence, because nobody gave me a role model with that same mindset.

That's a sad thing in some academic communities: until you prove yourself to be a good-enough specialist, you won't be respected as a good generalist. Schools expect little kids to show that they love spending the whole day on a small coding assignment and debugging it. If they don't love it, they are told they are not a good fit for computing. That's the message they hear from the teachers. But is it really a competency we need? Is computing only for those who want to spend all day debugging their assignment?

And by opening up the field in this way, it becomes attractive for more boys as well. Diversity is not about boys and girls. It's not about women and men. Diversity is about being able to see value in different mindsets and problem-solving strategies, and seeing value in people who are different—without first testing to see if they are just like we are. It means being able to see those other qualities, even if, at the moment, they are not as good as we are at this or that specific task.

In the end we want to become more open, because we need more people in tech. We need boys, we need girls, we need everybody who is interested,





they talk to universities? Some ability to be humble and listen to what the other side has more expertise in is the first step, and we are still not very good at it.

**Matej Hrušovský:** Have you been able to bring your interest in diversity to your role at the university?

**Barbora Buhnová:** Yes! Here at the university, the management of the faculty is really supportive of cooperation with Czechitas and supporting girls in their interest in tech. At the same time, however, there is some pushback from students who do not want anybody to get benefits that they are not able to get themselves. And now there is some pushback from students who see this as a way to disadvantage the male population.

We need to help people understand that we are trying to generate interest from all the talent reservoirs we have in the Czech Republic that might study computing. Studies from other countries are showing that activity to raise girls' interest in computing has beneficial effects on boys as well. So many boys have self-selected away from computing, possibly because they don't consider themselves specialists in technology. Now they see these activities for girls and think, "Okay, maybe I can also reconsider computing." The effect is beneficial on both sides.

**Matej Hrušovský:** Have you seen any significant increase in how many women come to study computer science or work at IT companies?

**Barbora Buhnová:** We are monitoring the numbers, so we can see the progress. There are some peaks, some up and down. When I was starting to study, there were roughly around five percent women in my class. Now it's roughly twenty percent women among students at our faculty. We are more lucky as a general university, because we have other faculties that make it more attractive for those multidisciplinary individuals. Technical universities have a bit of a harder situation, and the numbers are much lower there. The Czech Republic, according to Eurostat, has the lowest representation of women in the tech industry, at only ten percent among professionals. So I'm quite happy to see around twenty percent women in the classes at our faculty.

**Matej Hrušovský:** Where do you think the trend is going? Will it get to 50-50 representation?

---

Academic evaluations  
are changing, suggesting  
that having universities  
co-creating together with  
industry will be the way to  
excellence. There is no  
way to excellence while  
staying isolated from  
industry completely.

---

**Barbora Buhnová:** That's a very good question. I would say that the skillset needs we have in IT are changing. The ratio will depend on our ability to give credence to non-stereotypical skillsets, and that's a huge problem we have at the moment. For example, a UX designer who is not interested in programming has a skillset that is assigned a lower value. It's like we have this skillset that is a first-class citizen and then the skillsets that are second-class citizens.

Until we learn to value both of these skillsets because they are both important for the industry overall, people who don't have those stereotypical computing skills will leave the industry or hesitate to come in. If we learn to value all the skillsets we actually need in this industry, we can get very close to 50-50, but it may be a very long journey to rethink our notion of meritocracy.

Then there are many critics saying, "Look in those societies where they are very developed and women are free to choose anything. They still don't choose computing." And I ask them, "Why do you think they don't choose computing? Because they don't feel valued in computing." People will choose a field that values what they bring to the table.

I don't think reaching a 50-50 ratio is important. But when you have lower than thirty percent representation of a certain minority group, many people in that group will decide not to go into that industry. Some will

avoid that course of study because they don't feel they belong there. After you cross this threshold, people are choosing more freely.


To bring that back to the low percentage of women represented in tech in Czech Republic: I believe the industry is doing its best to change the message. The leaders of tech companies are trying. There is nobody putting up obstacles and saying, "You are a woman, you don't belong here." Still, many women do not go to the industry because they see that, right now, it is dominated by men, and they think, "I don't belong there."

**Matej Hrušovský:** Thank you for sharing your experience and knowledge on this topic. Is there anything else you think needs to be said?

**Barbora Buhnová:** In the end, it is about telling people who want to give tech a try to give it a try. If they need help, provide that help. Have more compassion for newcomers, because often girls take on the role of a newcomer. Whether it's in elementary school or at a university, boys often have a longer history with computers than girls have. And it's always difficult to be a newcomer.

Part of having compassion for newcomers is not trying to test them. Many people have a mindset like, "Okay, let's give her this question to see if she is really good." That's a toxic mindset many people in IT have towards those who are different. And so many

girls say they are getting tired of proving everybody wrong.

Instead, let's be grateful for anybody who wants to get into this industry and help us work together, because it's a big task and we don't have enough people. Let's make room for anybody who wants to give it a try. 

Diversity is not about boys and girls. It's not about women and men. Diversity is about being able to see value in different mindsets and problem-solving strategies, and seeing value in people who are different...



**Pictured:** The Czechitas team



# What is Red Hat developing **NEXT?**



Learn more at  
**next**.redhat.com



Feature



About the Author

**Ján Jančár**

is a PhD student supervised by Vashek Matyas at the Centre for Research on Cryptography and Security (CRoCS) at Masaryk University in Brno working on side-channel attacks, elliptic curve cryptography, and security of cryptographic implementations. His research focuses on security of cryptographic implementations of asymmetric cryptosystems, mainly against timing and other side-channel attacks.

# The need for constant-time cryptography

Timing attacks have been used successfully against a variety of popular encryption techniques, but they can be prevented with consistent use of constant-time code practice.

by Ján Jančár

Cryptography provides privacy for millions of people, whether by ensuring end-to-end encrypted messaging, [securing more than ninety percent](#) of the web behind HTTPS, or establishing trust behind the digital signatures used in electronic ID cards. And all of this cryptography needs to be implemented somewhere, by someone.

In the case of HTTPS, encryption is usually done in open source cryptographic TLS libraries such as OpenSSL, which is [used by most web servers in internet-wide scans](#). Other implementations, however, are closed source. For example, the smartcards used by electronic IDs are done by the vendors of the cryptographic hardware. These libraries and smartcards have suffered serious vulnerabilities in the past, mostly coming from memory safety or logic issues, but some vulnerabilities will be found not in the code, but in its outside behavior.

All code, including cryptographic code, has to be executed on actual physical hardware. This execution takes some amount of time, uses some amount of power, might produce results and errors, and, like any electrical device, it emits a small amount of electromagnetic radiation and heat. All of these results of the execution

depend, in one way or another, on the code that was executed and the data that the code used. The timing dependency on code executed is clear: executing instructions on a processor as well as accessing memory takes time. The code can also branch on data and thus execute different instructions, which will result in a dependency between execution time and data.

Modern processors have caches that store values recently used from memory for very fast access, with the idea that recently used memory addresses will be used again. Caches provide fast memory access only if the read address is present in them (a cache hit) and provide no speedup on a missing address (cache miss). This leads to some interesting behavior: the same sequence of instructions accessing the same memory addresses can take different execution times, depending on the state of the caches.

Digging deeper into how the logic gates and transistors in modern processors work reveals the dependency between code/data and power consumed by a device. Logic gates consume different amounts of power when their output changes versus when it stays constant. This changing power consumption directly leads to the dependency of power—as well as electromagnetic radiation and heat—on the executed code and data.



```
def check_password(password):  
    correct = "hunter2"  
    for i in range(len(password)):  
        if i >= len(correct) or password[i] != correct[i]:  
            return False  
    return len(password) == len(correct)
```

**Figure 1.** Example code vulnerable to a timing attack

The physical and temporal effects of computation, usually called side-channels, are clearly established. However, we also need to know whether they can impact security. Do we need to care about side-channels? Do we need to avoid them completely or at least minimize them?

These questions were answered twenty-five years ago in a [seminal paper by Paul Kocher](#), which introduced the concept of *timing attacks*.<sup>1</sup> In a timing attack, the attacker extracts secret information by measuring the execution time of a cryptographic function operating on secret keys or information.

A piece of code leaks secrets through a *timing leak* if it conditionally branches on secret values or accesses memory based on secret values. Kocher presented attacks on example implementations of several then-popular cryptosystems, such as Diffie-Hellman, Rivest-Shamir-Adleman (RSA), and Digital Signature Standard (DSS), variants of which are still used and attacked today. Timing attacks require that the attacker can measure the runtime of a sensitive operation, function, or call with some precision. The required precision and amount of

measurements vary significantly based on the size of the timing leak and noise.

A simple example of code vulnerable to a timing attack is displayed in **Figure 1**. Try to see for yourself how this code could be vulnerable, assuming that you as an attacker can trigger the `check_password` function with any input password and can measure its execution time precisely. The goal is, of course, to obtain the correct password embedded in the function, ignoring for a moment that passwords should be stored hashed with salt and pepper and not hardcoded. Stop and don't read further if you want to find the attack yourself, as its description follows.

The attack is an iterative one and is based on the fact that the early return in the `for` loop terminates the execution of the function at the first character of the input password that is different from the correct password. This allows the attacker to guess the password character by character from its beginning, trying "A\_" "B\_" etc., which will take a similar amount of time, until "H\_" takes longer.

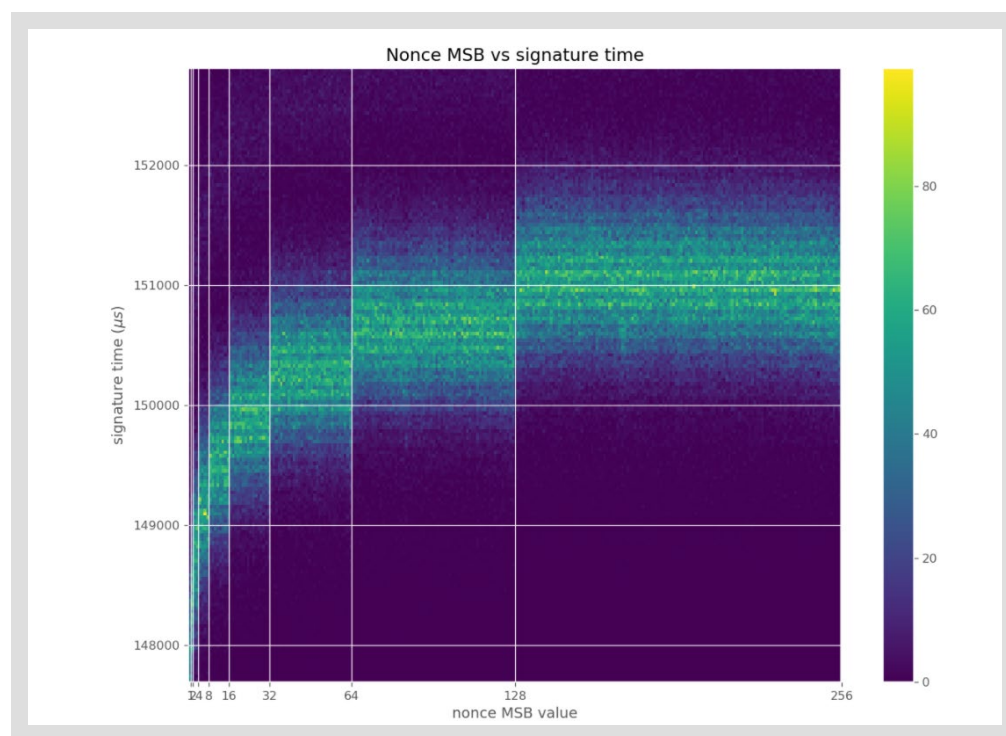
It might be surprising, but very similar attacks were used to target many popular

crypto implementations, including some recent implementations of post-quantum cryptography.<sup>2</sup> Their target was the **memcmp** function, which is used to compare buffers of data. **Memcmp** needs to be fast and thus similarly aborts the comparison as soon as it finds a difference. The attacker can then gain rough information about how many bytes were equal, which can be used to brute force a value compared against an attacker-controlled one. This was enough to mount a serious attack on several cryptosystem implementations that hope to win a competition of cryptosystems for a future world with quantum computers.

### THE MINERVA ATTACK

Our research first seriously dived into timing attacks with the **Minerva** group of vulnerabilities in the Elliptic Curve Digital Signature Algorithm (ECDSA). Elliptic curves are mathematical objects with rich properties that happen to be very well suited for building cryptography. Roughly speaking, an elliptic curve is a set of points on a plane, the coordinates of which satisfy an equation of the form  $y^2 = x^3 + ax + b$ . There is a very natural way to add points on the curve and obtain another point on the curve, in the same way that adding integers produces another integer. On elliptic curves, adding the same point to itself many times (i.e., multiplication) is called scalar multiplication to signify that one is multiplying the point with a scalar.

The similarities with the addition of integers end here. It is easy to invert integer multiplication by simply dividing with the integer multiplicand



**Figure 2.** Heatmap of the signature duration and nonce most significant byte value on the Athena IDProtect card

and obtaining the multiplier. However, inverting scalar multiplication on well-chosen elliptic curves—that is, computing the scalar used to multiply a point given the point and its multiple—is a hard problem for which there are no practical algorithms known. This hard problem forms the basis of elliptic curve cryptography and the ECDSA cryptosystem.

ECDSA is a digital signature algorithm. When a signer signs a message (a document, for example), they need, or rather their cryptographic implementation needs, to pick a random integer called a nonce. It then performs scalar multiplication of a

given fixed point with this nonce as the scalar, using the result in some further computations to produce the signature. The nonce needs to be uniformly random in a given fixed range and can never be revealed to the attacker or reused. Doing so would lead to a complete break of the cryptosystem, giving the attacker the capability to forge signatures. Even giving partial information about many used nonces to the attacker can have catastrophic consequences, as attacks using lattices can extract the private signing key and break the cryptosystem.

We discovered the first vulnerability that would come to be in the Minerva



group of vulnerabilities while working on [ECTester](#), a tool for testing black-box elliptic curve implementations in smartcards and software libraries. We first noticed an issue in a quick Python Matplotlib heatmap of the most significant byte of the random nonce and the runtime of ECDSA signing on the Athena IDProtect smartcard, which is displayed in **Figure 2**. If there were no timing leak, the figure would look like a horizontal line, showing no relationship between execution time and the value of the secret random nonce. However, the ECDSA implementation on the Athena smartcard was leaking the bit length of the random nonce linearly; that is, each additional bit in the binary representation of the random nonce added some processing time, which is visible in the figure as distinct steps at the powers of two values. With each measured ECDSA signing, the attacker thus gains noisy information on the bit length of the random nonce used in the signature.

Is this enough for an attack, given that the common ECDSA private key has 256 random bits and thus is uniformly distributed among  $2^{256}$  values? Luckily for the attacker, knowledge of the bit length can be turned to knowledge of the most significant bits of the random nonce (small bit length means that topmost bits are zero), and then a lattice attack can be used to extract the private key given enough measured signatures. We demonstrated this attack on the Athena IDProtect, which was both Common Criteria and FIPS

140-2 certified, and also enhanced the attack method significantly. After extending our data collection to include open source cryptographic software libraries, we found five more libraries with the same timing leak. Similar leakage was found in a certified Trusted Platform Chip by a [subsequent paper](#).

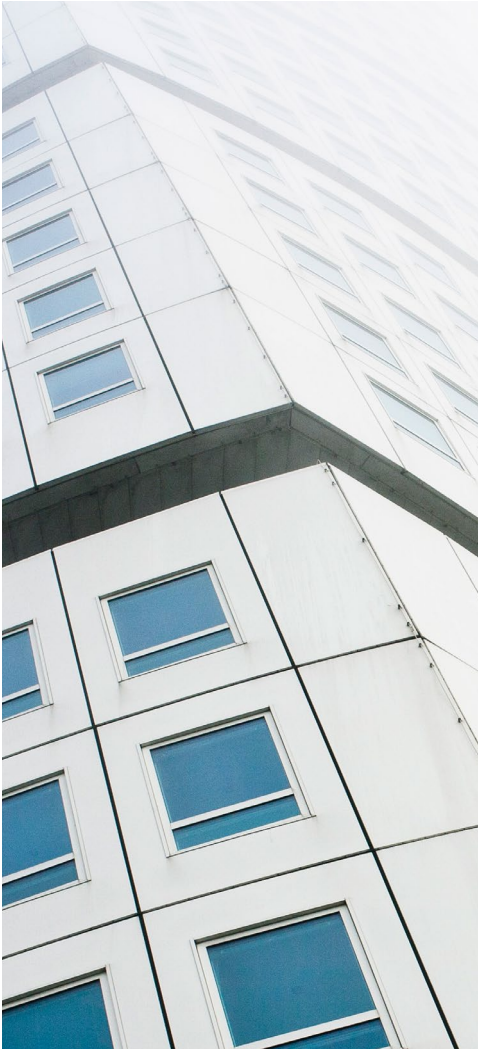
## DEFENDING AGAINST TIMING ATTACKS

What could the vulnerable implementations have done to prevent leaking sensitive information through a timing leak? The most common countermeasure against timing attacks is to develop sensitive cryptographic code with *constant-time code practice*. That is, to develop code such that:

- There is no information flow from secrets to branch conditions or loop bounds.
- Addresses used for memory access are not influenced by secret data.
- Secret data is not given as arguments to some instructions that might execute in variable time.

Following the above rules ensures that even an attacker who can measure the execution of the sensitive code and observe its memory accesses will not be able to gain any information about the secrets used. Some algorithms, like those for scalar multiplication on an elliptic curve, have versions that are naturally constant-time (i.e., that satisfy the above conditions, thus are safe from timing attacks). Other countermeasures

While 11 out of 15 popular cryptographic libraries test the correctness of their code in their continuous integration (CI) pipelines, only 4 of them use the tools to verify resistance against timing attacks in their CI.



<sup>1</sup>“Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” CRYPTO (1996).

<sup>2</sup>Qian Guo, Thomas Johansson, and Alexander Nilsson, “A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM,” *Advances in Cryptology—CRYPTO 2020*, pp. 359–86.

against timing attacks exist, for example, blinding, which randomizes values used inside the computation to make the leak useless. However, the constant-time code practice remains the most frequently used.

There is one significant issue with ensuring that code is constant-time while reasoning about it on a source code level, namely the compiler. Recent research has shown that modern optimizing compilers can easily take code that looks constant-time and produce a binary that is leaking. Developers thus often need to trick the compiler into not introducing these issues in their code.

If trusting developers to follow secure programming guidelines, including the constant-time rules, were enough, there would be no bugs or vulnerabilities in any code. As all humans make mistakes, we should not leave them alone, fighting in the trenches, but provide tooling that will help them make their code constant-time or test whether it is constant-time, similar to how testing and fuzzing are used to help ensure code correctness. Luckily, such tools already exist. In fact, a [recent analysis of the landscape](#) found over thirty of them. The tools range in functionality and approach: some take code and transform it into constant-time code, while others verify that existing code or binary is constant-time. Some tools use a formal approach and are sound, meaning that there is proof that they will find all timing leaks in the code.

## WHY ARE TIMING ATTACKS STILL AROUND?

Given that there are effective countermeasures, techniques, and tools for verification of resistance to timing attacks, why are there still new timing attacks? To begin answering the question, we can look at the use of tools for verifying resistance to timing attacks. While eleven out of fifteen popular cryptographic libraries test the correctness of their code in their continuous integration (CI) pipelines, only four of them use the tools to verify resistance against timing attacks in their CI. Why are these tools not widely used? Is it that developers don’t know about them? Or is it the typically poor usability of tools that are mainly research artifacts and not ready-to-use products? We hope to answer all of these questions in our research.

To do so, we decided to ask the developers directly by conducting a survey with developers of prominent open source cryptographic libraries in cooperation with researchers from Max Planck Institute for Security and Privacy in Bochum, Germany, and the University of Rennes, France. The survey concluded with forty-four developers from twenty-seven libraries participating and sharing their knowledge and opinions on timing attacks, countermeasures, and tools. We hope that this study gives us interesting insights into developers’ views towards timing attacks—and that we can move from arguing about timing attacks using anecdotal evidence to discussions based on proper results. Expect a paper with details soon. 

# The elastic bare metal cloud is here

Exclusivity of resources is becoming obsolete.

The Elastic Secure Infrastructure Project (ESI) provides a solution for sharing computing resources and getting the most from hardware investments.

by Gagan Kumar

Using resources efficiently is an important goal for any organization. If those resources are computers, then theoretically that goal should be easily achievable, because machines don't get tired and need a break. In the real world, however, job scheduling is complex, and resources will likely be sitting idle at times. What if an organization could lease their idle servers when they were not being used, and, for this favor, be compensated with access to other external servers at times when the organization needs them? The Elastic Secure Infrastructure (ESI) project is pursuing this idea to create more options for server efficiency.

## EFFICIENT AND ELASTIC BARE METAL CLUSTERS

[Mass Open Cloud](#) (MOC, massopen.cloud), one of several research initiatives Red Hat Research supports, is a partnership of universities, research institutions, and industry working to create an open public cloud exchange model. MOC needed to create a bare metal cloud comprising hardware from

---

ESI allows communities to build a more diverse selection of servers than commercial services can support, and it also allows building a bigger community around IT research and development.

---

multiple organizations involved in MOC, but the organizations also wanted to maintain complete control over the servers they own. Then, if the servers were idle, they would have an option to lease their hardware to other parties who might need it to create their own private network to form a cluster.

Organizations and departments in the IT industry build dynamically scaled services, and, to maximize efficiency, they want to be able to lease unused IT infrastructure to others, as long as they can claim it back when the need recurs. MOC created projects such as [HIL](#) (Hardware Isolation Layer) and [M2](#) (malleable metal) to achieve this goal. However, scaling this service required a lot of engineering effort

and upstream changes in OpenStack. This effort led to the creation of the Elastic Secure Infrastructure project.

Using ESI, if a bare metal server is not currently in use and isn't required immediately, the owner can lease the bare metal machine to a lessee who can use the machine for their need for an agreed amount of time. ESI enables organizations to build bare metal clusters that are elastic enough to cater to an organization's changing computing needs.

## BRINGING MULTITENANCY TO IRONIC

The ESI project is built as a solution on top of [OpenStack](#). [Ironic](#), a service in the OpenStack platform, already



existed for managing bare metal machines, but it assumed that all computers managed by IroniC were owned by a single administrative organization. The ESI project proposed to add multitenancy to IroniC, and began working with the upstream developers. We helped to develop node multitenancy so that users could manage their own bare metal resources without requiring global administrative privileges.

While researching IroniC, we found an existing IroniC field called Owner that was used only for informational purposes. We proposed using this field for access control as well. In most of our use cases, owners leased the nodes to nonowners for a specific time period. We wanted to keep track of this information, so we also added another field called Lessee. The final step in our work was to create policy rules for owners and lessees, through which administrators can expose the node API calls to node owners and authorized lessees, based on updating policy files. These developments were the foundation for a shared bare metal cluster. (Read more about the development of multitenant functionality for IroniC in [“Isn’t multitenancy ironic,” RHRQ 2:2, August 2020.](#))

## ESI, VIRTUALIZATION, AND PUBLIC CLOUD PROVIDERS

At this point, you might be wondering, “Why can’t we use virtualization instead of ESI?” Virtualization can address some resource sharing requirements. However, some

---

Software that allows groups to build services across silos also increases flexibility for research teams and can reduce the total cost of ownership compared to purchasing resources by the minute for the overall organization.

---

application performance and isolation requirements can only be addressed by using the capabilities of an entire bare metal machine, without the middle layer of virtualization between hardware and software. Furthermore, some organizations need to share hardware between projects in a more secure way than virtualization will permit. The ESI project can better address both of these requirements.

Another question that the ESI team frequently gets asked is, “Why should we use ESI when we can rent a server from any public cloud provider?” ESI allows communities to build a more diverse selection of servers than commercial services can support, and it also allows building a bigger community around IT research and development. Currently, much of the software that commercial providers use to manage servers at lower hardware levels is completely opaque to users and developers. With a more diverse community, users may be able to get access to hardware that they wouldn’t otherwise be able to try, such as new chipsets and other hardware under development with partners who participate in the leasing service to gain early user feedback.

Research IT departments in academic and industry organizations are especially interested in sharing their resources and collaborating with other departments, universities, and research teams. Software that allows groups to build services across silos also increases flexibility for research teams and can reduce the total cost of ownership compared to purchasing resources by the minute for the overall organization. The ESI project is in the process of making these collaborations easier. If you are in a situation where your bare metal machines are idle for a long period of time and you want to participate in a cloud marketplace for using the machines more efficiently, or if you are looking to find specific types of bare metal machines you currently can’t access, the ESI project could be your solution.

## IMPORTANCE OF OPEN SOURCE IN ESI

Trustworthiness is essential for any shared service. In the ESI project, open source development has played a major role in developing trust in the community. The code isn’t a black box: a potential owner can go through the entire process and all the code before committing their resource to

the ESI leasing platform. We are also continuing to work with upstream groups to add more security options, such as software attestation, to ESI. The open source model likewise facilitated development when we identified multitenancy as a feature gap in Ironi and worked with the upstream community to develop new code. This work wouldn't be possible if Ironi were a proprietary service.


## PILOT DEPLOYMENT

After several years of early research and development, the ESI concepts are no longer theoretical. The ESI engineering team successfully deployed a pilot environment in the Massachusetts Green High Performance Computing Center ([MGHPCC](#)) in 2021. Our goal is to support leasing functionality for bare metal machines in the [Open Cloud Testbed](#), Mass Open Cloud, and [CloudLab](#). We encourage other datacenters to create their own leasing federation or join an existing leasing federation and contribute to this project.

## WHAT'S NEXT FOR ESI?

The engineering team of the ESI project is now planning to support the installation of software that is expected to interact with the leased bare metal servers. We are in the process of identifying other feature gaps that need to be addressed to run platforms such as Red Hat® OpenShift® on top of the bare metal machines leased from the ESI service. Once this feature is supported, it could fundamentally change the way OpenShift is

administered, because nodes in the infrastructure could also become elastic based on the platform workload.

The ESI team is also exploring the possibility of building software to support a marketplace for the leasing service and a credit system for administrators who maintain an ESI service. Once ESI starts to scale horizontally, it becomes crucial to provide users the ability to post their available hardware in a marketplace and for the lessees to present various options for how they want to use services. A team of Boston University students and Red Hat interns produced a proof of concept for a bare metal marketplace called [FLOX](#) and presented it to the audience as part of DevConf.us 2019. The ESI team is continuing this work. 



## About the Author

**Gagan Kumar** is a Product Manager in the Red Hat Research team. He has a master's degree in computer science from Northeastern University, Boston. Gagan's primary focus is on Red Hat's engineering activities related to Mass Open Cloud and the Operate First initiative. He is working on projects that are related to bare metal sharing systems and metrics collection for OpenShift instances.

## ABOUT THE PROJECT

**The Elastic Secure Infrastructure initiative is an open source project.**

We appreciate as much involvement as possible in the process of developing the project. If you want to be part of this team, learn more about our road map, or just see a demo of the project, you can email the ESI team at [esi@lists.massopen.cloud](mailto:esi@lists.massopen.cloud).

If you want to clone the project's GitHub repository, or submit a feature request, you can do so by visiting our [GitHub repository](#).

**You can also contact the ESI development team on the OFTC #moc IRC channel.**

## Feature

## Creating a Linux-based unikernel

Is there a way to gain the performance benefits of a unikernel without severing it from an existing general-purpose code base? Boston University professors, BU PhD students, and Red Hat engineers at the Red Hat Collaboratory at Boston University are getting close to finding the answer.

**About the Author****Gordon Haff**

is Technology Evangelist at Red Hat, where he works on emerging technology product strategy, writes about tech trends and their business impact, and is a frequent speaker at customer and industry events.

His books include *How Open Source Ate Software*, and his podcast, in which he interviews industry experts, is *Innovate @ Open*.

by Gordon Haff

A unikernel is a single bootable image consisting of user code linked with additional components that provide kernel-level functionality, such as opening files. The resulting program can boot and run on its own as a single process, in a single address space, and at an elevated privilege level without the need for a conventional operating system. This fusion of application and kernel components is very lightweight and can have performance, security, and other advantages. An ongoing Red Hat Collaboratory research project is creating a unikernel that builds off Linux with relatively few code changes.

### HOW WE CAME TO LOOK (BACK) AT UNIKERNELS

The basic unikernel concept is not new. Although unikernels are often traced to research projects like Exokernel and Nemesis in the late 1990s, they're similar in concept to the very first operating systems on early 1950s mainframes.

Early batch-processing programs were statically linked to shared libraries to perform tasks

like managing buffers and input/output (I/O) devices. One of these early libraries, the SHARE Operating System, written by the SHARE User Group for the IBM 709 in the late 1950s, was an early example of users sharing code that they wrote among themselves—in effect, early open source. Yes, users wrote some of the earliest operating systems for themselves. All code ran at a single privilege level, from start to end, without multitasking or timesharing; there was no scheduler. In other words, it was a unikernel—even though the term wasn't coined until much later.

In part due to the cost of early computers, operating systems evolved over time and acquired additional functionality. Timesharing was one particularly important development in the 1960s; eventually it largely supplanted batch processing. One major milestone, released in 1969, was Multics, a collaboration between MIT, Bell Labs, and GE. Although the project was mostly a failure, in part because of its complexity, it influenced a wide range of minicomputer operating systems, like Digital Equipment's VAX/VMS as well as Unix, which came out of Bell Labs.



What if we could  
take an open source  
operating system with  
a large community,  
Linux, and add  
unikernel capabilities  
into the same source  
code tree?

Multics introduced innovations like dynamic linking and hardware support for ring-oriented security, which are familiar components of modern operating systems, including Linux.

These modern operating systems are extremely sophisticated and capable. But might we reexamine simpler approaches? For example, with a unikernel, the complex boundaries and permission checking a standard, multiuser operating system requires are no longer needed.

The unikernel program model now makes a lot more sense than it did back in the days of large, expensive computers. Today we have very inexpensive multiprocessor/hyper-threaded CPUs and the potential for virtualized environments running on a host that supports hundreds or even

thousands of guests. We no longer have to be concerned about wasting compute cycles if a unikernel image blocks for I/O and does not context switch to another process. However, since a unikernel image can and does support multithreading, context switching between threads is viable.

#### CAN LINUX MAKE IT SIMPLER?

The usual approach to building a unikernel is either building a specialized operating system from the ground up or forking an existing operating system, removing components, and modifying it as needed. Both approaches require ongoing maintenance of the resulting unikernel— which, among other problems, makes it harder to benefit from continuing enhancements to a general-purpose code base, including support for new devices.

Unikernels can also miss out on performance gains from new types of hardware acceleration, working against a key motivation for developing a unikernel. Furthermore, unikernels can require application changes, and they may not support the POSIX standard. Custom toolchains may be needed.

However, what if we could take an open source operating system with a large community, Linux, and add unikernel capabilities into the same source code tree? After all, Linux already supports a wide range of architectures and can be built in different ways depending upon the target use case.

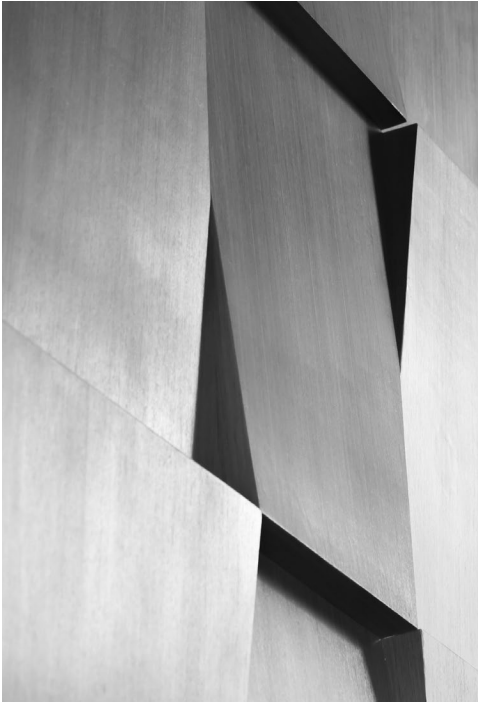
That was the question a collaboration of professors, PhD students, and engineers at the BU Red Hat Collaboratory set out to answer.

---

#### The team set four goals:

1. **Ensure** that most applications and user libraries can be integrated into a unikernel without modification. Building the unikernel should just mean choosing a different GNU C Compiler (GCC) target.
2. **Avoid** any ring-transition overheads. Overhead experienced by any application requesting kernel functionality should be equivalent to a simple procedure call.
3. **Allow** cross-layer optimization. The compiler and/or developer should be able to co-optimize the application and kernel code.
4. **Keep** changes in Linux source code minimal, so they can be accepted upstream and the unikernel can be an integral part of Linux going forward. This will ensure unikernels are not an outsider but a build target for which anyone can compile their applications.

They wanted to meet these goals while continuing to support complex applications and a rich hardware compatibility list (HCL)—and while preserving the familiar configuration and operations model, as well as the debugging and optimization capabilities



of the OS. And do all this without impacting other build targets. And, finally, do so in a way that enables, over time, the performance optimizations that have been demonstrated by other unikernel researchers.

The team examined a number of options but decided to avoid approaches that involved significant application rewrites, allowed arbitrary applications to run alongside the kernel in the ring 0 privilege level, or required one or more components running in userspace. They settled on a pure unikernel approach, whereby the kernel is statically linked to run a single application.

## A UNIKERNEL LINUX (UKL)

A prototype came together fairly

quickly, with minor changes to the Linux kernel. After making the code changes, they created a prototype by building the Linux kernel with a UKL config option turned on. The linking stage in the kernel build process was slightly modified to link object files created from the GNU C Library (glibc), the application code, and a UKL library. The prototype served as a proof of concept, and a simple benchmark validated resulting performance gains. The work was presented in “Unikernels: the next stage of Linux’s dominance” at HotOS ’19 in Bertinoro, Italy. Since that time, the team has continued to build on the initial work.

UKL builds an unmodified application into a Linux-based unikernel; it runs in the same privilege level as the kernel (ring 0) and allows for many optimizations. It consists of a small set of changes to the Linux kernel (less than 1,500 lines of code), which allows UKL to use Linux’s well-tested code base and work in concert with the large, established Linux development community rather than doing a standalone project.

UKL largely supports the POSIX interface. Differences are in two specific areas:

First, UKL runs as a single process. Therefore, `fork()`, which causes a process to make a copy of itself, doesn’t make sense in a UKL context. However, UKL does support `clone()`, which creates a new thread. This allows the entire POSIX threading

library (libpthread), which is central to concurrent process flows, to work.

Second, the application cannot make an *explicit* syscall. Instead, the far more common case of syscalls being used behind the scenes to have the kernel perform some privileged task is handled by the modified glibc library. Changes to glibc largely mask the fact that the linked application is now running in ring 0 rather than userspace (ring 3), where it would be running in the case of stock Linux. The modified glibc makes an operation such as opening a file, `open()`, simply call a kernel function rather than first transition from ring 3 to ring 0 and then transition back, along with the associated stack operations.

The build step is straightforward, which is often not the case with unikernels. Typically, unmodified applications are rebuilt and linked with the modified glibc. Then UKL is built as you would build Linux normally. Its final linking step takes in the partially linked user binary and creates a `vmlinux` that can be deployed anywhere. (`Vmlinux` is a statically linked executable file that contains the Linux kernel in one of the object file formats supported by Linux.)

There is no custom toolchain, although the researchers hope to encapsulate these steps into a single make step in the future.

## RUNNING UKL

Because UKL inherits Linux’s large HCL it can run in either a virtual machine or on bare metal. When UKL

boots, it starts running the workload. Optionally, you can build a UKL to have a sidecar. With this sidecar, normal user space applications can run alongside the UKL main workload. This allows you to run a shell or other utilities to manage the system or debug it, for example. All the tools normally used to debug Linux can be used with UKL. These utilities run in user mode, as they would on normal Linux.

---

UKL gave 23% tail latency improvement and 33% throughput improvement over the Linux baseline.

---

Because the UKL workload runs in kernel mode, it has access to all the internal kernel functions. This provides the ability to occasionally bypass kernel code entry/exit and invoke the underlying functionality for performance improvement. The research team has also tested versions of UKL that have no stack switches upon kernel code entry exit; doing so has also provided performance benefits. Additional performance tweaks came from manually shortening the tcp recvmsg/sendmsg paths in the kernel and calling these from network-based UKL workloads.

To date, the biggest performance boosts came from a workload containing the Redis database. UKL gave 23% tail latency improvement

and 33% throughput improvement over the Linux baseline.

### THE BIGGEST CHALLENGES

In keeping with the researchers' goals, with UKL syscalls become simple kernel function calls, without involving ring transitions back and forth to kernel space. However, eliminating that ring transition has presented some of the greatest challenges associated with UKL.

The first challenge relates to differences between a normal user stack and a kernel stack. Normally, a page fault occurs when a process accesses a page that is mapped in the virtual address space but is not loaded in physical memory. These aren't normally errors and are used to increase the amount of memory available to programs in Linux that use virtual memory.


However, when running in ring 0, the hardware does not switch stacks on a page fault. When state is pushed on the stack, a double fault results—and the system crashes. UKL addresses this for now by ensuring that pages are mapped ahead of any operations that could result in a fault. So the solution for the double fault issue is either preventing it by pre-faulting the stack before entering the kernel or switching to a wired kernel stack when a double fault does occur.

Another challenge is that during the normal transfer back from ring 0 to ring 3, the system does a great deal of post-

processing in areas such as I/O, signal handling, and read-copy-update (RCU) synchronization. Not doing so caused a significant performance hit. As a result, UKL simply added calls to the kernel functions that deal with these housekeeping details. Subroutine calls are made to the existing system calls rather than using syscall instructions in the absence of a ring change.

### WORK CONTINUES

The BU researchers and Red Hat engineers working on this project see a variety of opportunities to continue improving the performance of complex concurrent workloads. Because the application is running in kernel mode when using a unikernel like UKL, there are many opportunities for synchronizing certain operations in ways that are difficult for user space code to accomplish.

Of equal or even greater importance, however, is working with the Linux community to get UKL code into upstream Linux. The proposed changes are relatively few and non-invasive, which should make inclusion easier. This would allow for a unikernel that both benefits the Linux community and gains the benefits of an open source development model. 



*The author would like to thank BU PhD candidate Ali Raza and Red Hat Senior Distinguished Engineer Larry Woodman for their invaluable assistance with this article.*



Feature



About the Author

**Marek Grác**

is a senior software engineer and researcher at Red Hat's Brno office. His interests and expertise as a teacher are in machine learning, DevOps, and programming courses, and he has supervised dozens of student theses.

## Making machine learning accessible across disciplines

Machine learning has been driving research breakthroughs in many fields. Now there is an open source curriculum designed to help non-specialists build the skills they need to use it.

*by Marek Grác*

**M**achine learning is an increasingly important competency in a growing number of fields. Biochemists are using it to create models for protein engineering. Economists are using it to predict the price of cryptocurrencies. And linguists have applied it to the problem of fake news. Computer science is not traditional domain knowledge for these disciplines, but massive data processing is often a requirement for cutting-edge research in natural sciences, social sciences, and the humanities.

That is why we created the Applied Machine Learning course at Masaryk University, in Brno, Czech Republic: to help students outside the field of computer science learn more about machine learning without making them into computer scientists first.

### FOCUSING ON WHAT STUDENTS NEED TO KNOW

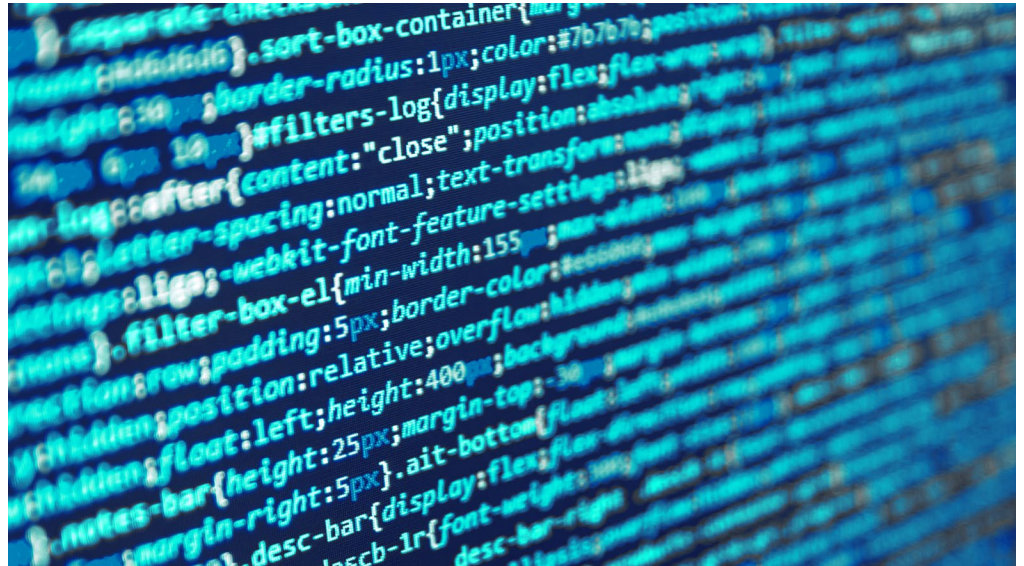
The field of machine learning is changing on a daily basis. Keeping up with trends is difficult, even for computer science professionals. Interested laypeople have two basic choices. They can join

one of many massive online courses, but they will have support only from peers for guidance. They will likely also be learning in isolation, separate from their local community. For those who are still students, they can join one of the courses provided by their university's computer science department. There is a catch, however. University courses are good for learning about theory and background. Modern technologies and tools, however, often develop too quickly to be added to syllabi while they are still new. In the context of machine learning, "modern" might be just a few months old.

We designed the Applied Machine Learning course to fill the gap between a theoretical and a practical approach. We wanted to show students that they can work with machine learning without the ability to rewrite the complex methods currently used by software professionals. For the majority of learners, understanding basic concepts and being able to use machine learning methods is enough for their needs.

Our target audience for the course is not students of computer science but future experts

We wanted to show students that they can work with machine learning without the ability to rewrite the complex methods currently used by software professionals. For the majority of learners, understanding basic concepts and being able to use machine learning methods is enough for their needs.



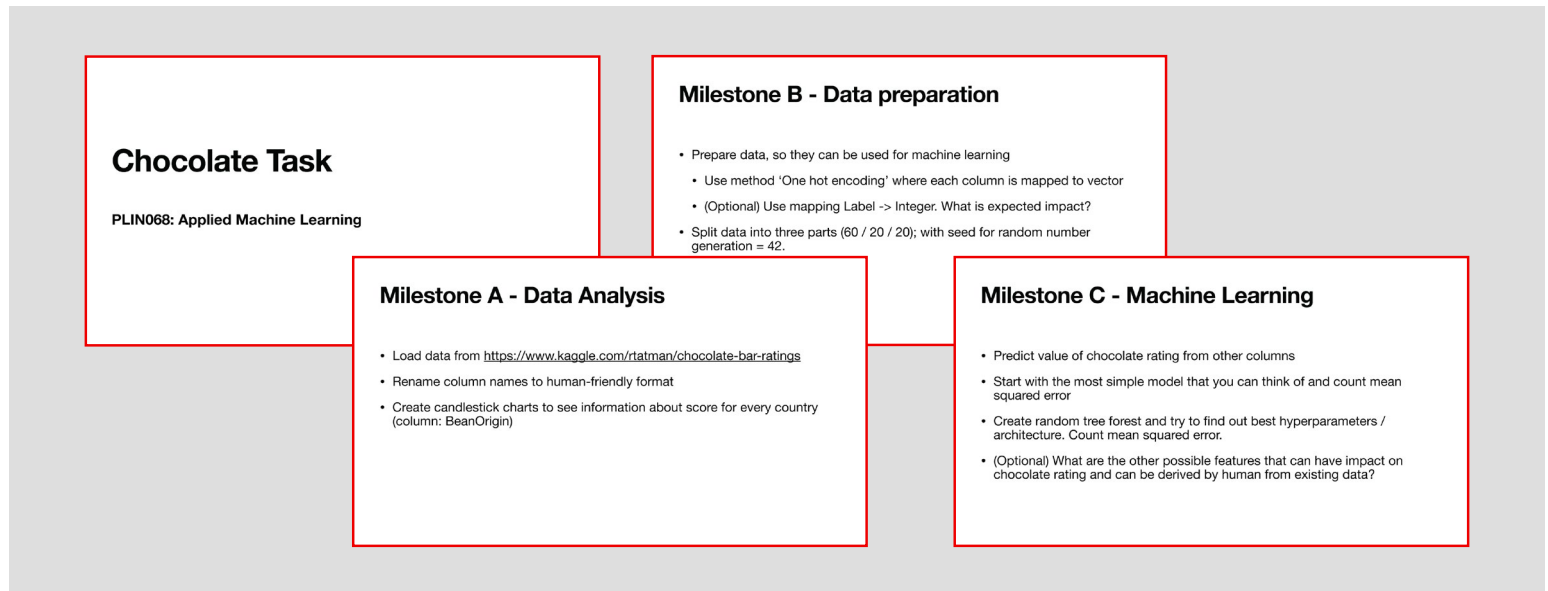
in different disciplines. To advance in their fields of study, it is important for these students to understand the capabilities of machine learning and be able to communicate with machine learning experts. Because machine learning is about correlation, not causation, domain knowledge helps us to distinguish between them correctly. In our first run, we were able to attract students from computational linguistics, computer science, chemistry, biology and economics. In fact, the use cases mentioned above were taken directly from these students' projects.

#### **ADVANTAGES OF AN INTERDISCIPLINARY CLASSROOM**

Putting all of these students together created a special environment. Our online, synchronous sessions were interactive, in small groups that were randomly shuffled before each task. By working in frequently changing groups, students were forced to

talk differently so students from other disciplinary backgrounds could understand them, whether they were talking about machine learning or their domain expertise. For their projects, we put students in groups of three, and each group purposely contained students from at least two different disciplines. Because the students had different backgrounds and goals, they could bring different views to the problem they were solving.

We wanted to give students an opportunity to learn new things without getting into a routine of simply opening an editor and coding something, so we had students creating annotation guidelines, fixing bugs in existing source code, trying to create better rules for simple games, creating chatbots, or trying to figure out which parameters might influence whether a person will get a mortgage. The different types of tasks allowed students to play multiple



**Pictured:** Student task slides (chocolate exercise)

roles, so they were not experts all the time. They needed to help each other in order to understand and complete the tasks.

Practicing routine coding and gaining knowledge of the libraries (e.g., Scikit, PyTorch, Keras) are an important part of the course, but those skills do not have to be exercised in groups. Thanks to our partner, [DataCamp](#), we were able to give students access to relevant routine exercises. These exercises allow students who don't fully understand the science of how coding works to learn the craft of writing in a programming language. And students made good use of them—on average they have spent more than 40 hours on this platform.


This course environment, along with COVID constraints, led to an approach where the teacher was more curator

than creator. We put together a set of videos for relevant topics from existing resources, including not only prestigious universities but also people from industry or even amateur science bloggers. Thanks to this approach, every week was presented by a different speaker, which helped us avoid the stereotypical instructor-centered—or “sage on the stage”—type of class.

## OPENING THE COURSE TO ANY UNIVERSITY

Our pilot group had only fifteen students, but it is not efficient to create such complex scenarios for a small group if there will be major changes every year. That is one reason why we open sourced all our materials on GitHub, with links to videos, our group tasks, and a methodological explanation of why they were selected. And of

course feedback from students should lead to improvements in future courses.

We would like to open our “franchise” to other students, so if you believe that your local university might be interested, we will be happy to support them and work together on these study materials to make them better, just as we have done with open source software for so long. 



*I would like to thank Z. Hladká and D. Hlaváčková, who offered to let us create such a course at the Faculty of Arts. We would also like to thank Hugh Brock, who supported us from Red Hat Research. It would not be possible without their belief in our idea.*



## How COVID made our world smaller

RHRQ interviewed Idan Levi, the Research Interest Group leader in Israel, to find out how research collaboration has changed over the last year and a half as the world went virtual.

**RHRQ:** As the leader of Red Hat Research in Israel, you work with universities that are geographically dispersed, for example, Technion to the north, Ben-Gurion University to the south, plus the Interdisciplinary Center (IDC) of Herzliya and the Academic College of Tel Aviv-Yaffo closer by. How have those relationships changed since you started working more virtually?

**Idan Levi:** When I started my position in 2019, we felt that maybe some of these places were too far away. The norm for building relationships at that time was to meet face to face, see the people, then work with them hand to hand. Now, it's a whole other experience.

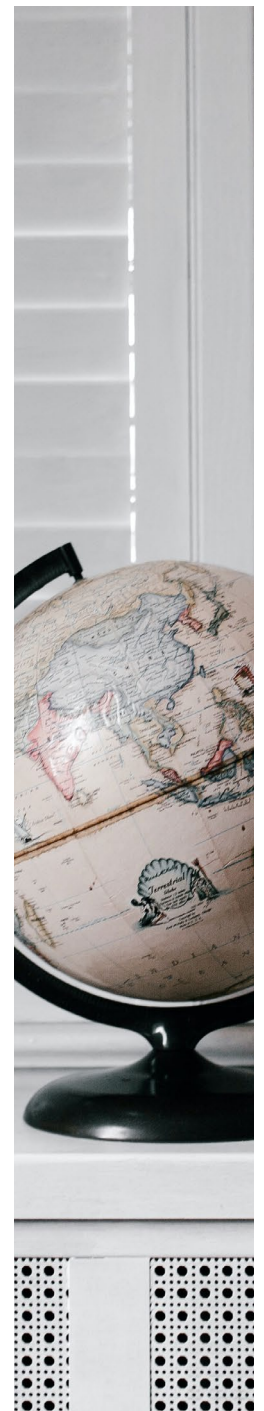
Obviously that creates challenges, but we were able to get some good things out of a bad situation. Being pushed towards this virtual medium made everything so close—everything is just a click away.

That was a huge driver that allowed us to build connections in a whole different way. For example, we've just started discussing an opportunity to collaborate with professors at Ben-Gurion University. That conversation started with someone from the engineering department saying, "Hey, there's this very talented master's student looking to do something with Red Hat to get some practical

experience." So I got to talking with him. He's from this remote institute. Right now the student doesn't have enough time to work with us, but it turned out that he was associated with the system engineering lab. That lab is very interesting to us, so we started to talk with his professors as well.

In normal times, we might not have been agile enough to make this connection and bring all the relevant people together. One of our engineers is living in the very far northern part of Israel and rarely comes to the office, two other engineers are in the office, and the professors need to travel from the university, which is ninety minutes south of us—and somehow we need to sync everyone. All of this was challenging to schedule, but it became much simpler once we started working virtually. So this was a huge success.

Another challenge was that, at least in Israeli culture, we are very warm and loud; we like to hug and touch the people we are talking to. It was very hard to get used to working without that kind of connection. Also, we love to argue, to jump in the middle of a conversation. So maybe the virtual space forces us to have better manners! It's a very interesting experience. I can testify for myself, I have had to become a little bit more patient.





**RHRQ:** You also work with colleagues at other research centers, for example in Brno, Czech Republic. Has that changed too?

**Idan Levi:** I think the connection between the Israel RIG and Brno RIG has strengthened during this time. We just met more frequently, started to work more together, and we could understand what the others were doing and going through.

I believe that started because in the team meetings we started to share a little more about day-to-day life. We could not meet people face-to-face anymore, we couldn't get out of the house. So with our counterparts in Brno, we shared about everyday things in addition to conversation about the challenges that we face. We tried to learn from each other, because all of us were trying to learn how to adjust to this new norm. What is going wrong? What can we improve? We started to ask one another, "So, how do you handle that?" Those conversations build relationships, when we can say, "Maybe you have a better idea of what the hell is going on!"

**RHRQ:** What about your education initiatives? Naturally they became virtual, but were there any surprises there?

**Idan Levi:** When COVID hit, it was like, "What the hell are we going to do now?" Then we had a conversation with IDC Herzliya and they said to us, "We usually don't have summer courses, but this summer we might do a course because our students will not be taking a vacation abroad or flying back home to their families as they usually do. Do you have something for them? Maybe you could teach them something about open source."

The engineers who lead the Red Hat Beyond initiative, Irit Goihman and Liora Milbaum, jumped at the challenge. We had amazing results

---

It's more about rediscovering and relearning how to utilize virtual meeting spaces and make the most of them—things like having a virtual meeting culture, screen sharing, joint virtual boards, and other tools.

---

from that, using purely Zoom and Slack. Again, it's a challenge that you cannot really see the people, but it was very successful. (Read more about Red Hat Beyond in the article "[Combining experience with passion inspires a new mentorship program](#)" in issue 3:1 [May 2021] of RHRQ.)

We helped sponsor another great initiative called [ROSE](#) (Red Hat Open Source for Education), which is for high schools. This initiative takes students from an Arab city in Israel, Tira, and from a Jewish city, Ra'anana, and brings them together in one classroom to engage them. They teach basic Python programming.

When COVID hit, we helped them with some resources to transform the program from physical to virtual. Because they didn't know whether it was going to work, they offered the course to friends and family of employees for a trial run. We even had an employee who brought in her father to learn Python, so there was some interesting integration of age groups. The program was great in terms of education, but we still haven't been able to solve the integration part of it. That's an important problem to solve, as the aim of the program was to use the learning experience to bring people from different backgrounds closer together. So we are still facing

the question, “How do we create integration in the virtual world? What does that look like?”


**RHRQ:** The first Tel Aviv Red Hat Research Day also turned out to be virtual. How did that go?

**Idan Levi:** I think we were the one of the first to be the guinea pigs for that. It was challenging, because it really changed the whole dynamic and everything we thought we were aiming to do at a research day. We had planned it to coincide with the local Red Hat Forum, which includes interactions with customers, partners, and open source contributors. Without any in-person events, we needed to change the entire dynamics. What is the value proposition now? We had to rethink everything. Just learning how to use the platform in the correct way took time.

Since then, we’ve had two virtual events. The first event was particularly challenging, but in terms of participation it was a great success. We had high participation. The second event was much, much better in terms of planning, interactions and materials, and we had much better engagement with that event. People asked questions and talked and tried to understand further. It was challenging, but it worked beautifully.

**RHRQ:** Are there any practices or new means of collaboration that you adopted that you will continue using once COVID restrictions have passed?

**Idan Levi:** I don’t think it’s about using anything new. It’s more about rediscovering and relearning how to utilize virtual meeting spaces and make the most of them—things like having a virtual meeting culture, screen sharing, joint virtual boards, and other tools. It also opens up opportunities for others who are more shy, or who work differently, and gives them another way to participate. I think this is huge.

Our work schedule has also changed to include many more meetings with different teams and different locations, so we can include someone from Spain, or Italy, or wherever they are. After the experience we’ve gained, that communication is much more fluent and natural than in the past. I believe these things will stay with us as we try to come back to some kind of normal. 



**Pictured:** Idan Levi



## Project Updates

## Research project updates

Each quarter, *Red Hat Research Quarterly* highlights new and ongoing research collaborations from around the world. This quarter we highlight collaborative projects in the Czech Republic, at Masaryk University (Brno), Brno University of Technology, and Czech Technical University (Prague).

Contact [academic@redhat.com](mailto:academic@redhat.com) for more information on any project described here, or visit the [Red Hat Research Project Directory](#) on our website.

### **PROJECT: AUFOVER** (Automated Formal Verification)

**ACADEMIC INVESTIGATORS:** Honeywell: Mgr. Tomáš Kratochvíla (Masaryk); Paradise: Prof. RNDr. Jiří Barnat, PhD (Masaryk); Formela: Doc. RNDr. Jan Strejček, PhD (Masaryk); VeriFIT: Prof. Ing. Tomáš Vojnar, PhD (BUT), and Ing. Aleš Smrčka, PhD (BUT)

**RED HAT INVESTIGATORS:**  
Kamil Dudka and Ondřej Vašík

At DevConf 2021, we presented our innovative solution to fully automatic dynamic analysis of unmodified source RPM packages. The solution was based on csexec, which is a wrapper of the system's dynamic linker. Now we are extending this solution for select formal verification tools, namely CBMC, Divine, and Symbiotic.

Thanks to our experimental csmock plug-ins, developers can easily evaluate these tools on a given source RPM package by running a

single command in their terminal. Although we are not able to complete the formal verification for the majority of Fedora RPM packages, the csmock plug-ins are optimized to deliver at least partial results in a predictable amount of time.

We are still working with developers of the formal verification tools to make their output format easier to understand by developers. We have provided patches to record and report the names of binaries executed, command line arguments passed to these binaries, and absolute paths to files containing the source code.

In our test suite, multiple tools are exercised by the same set of tests to help us discover bugs in the formal verification tools themselves. Called aufover-benchmark, the suite is now available in a public git repository and covered by a publicly available continuous integration (CI) system. All projects developed as part of the AUFOVER project are now consolidated under a single namespace in Github ([github.com/aufover](https://github.com/aufover)) and Fedora COPR ([copr.fedorainfracloud.org/groups/g/aufover/coprs](https://copr.fedorainfracloud.org/groups/g/aufover/coprs)).



#### **Pictured:**

The Vega Project rack, nicknamed "REK"

## PROJECT: Innovation Scorecard

### ACADEMIC INVESTIGATORS:

Doc. Ing. Ondřej Žižlavský, PhD, Eddie Fisher, and Tetyana Shpilka (BUT)

### RED HAT INVESTIGATORS:

Marcel Gazdík, Vojtěch Sokol, and Tomáš Meszároš

The practical application of our Innovation Scorecard concept in three case studies within Red Hat has now been successfully completed.

The primary focus of the first two case studies was limited to agile software development. This focus was ideal for our team to check how valid and reliable the developed concept of measuring the success of any change or innovation had been. The first case study, known as Atomic Host (container automation), allowed us to review the initial theoretical concept and get it into better shape for the case studies that followed. The second case study, known as CI, was more challenging due to the technical complexities associated with continuous integration. This opportunity drove the need to adjust the initial concept so it could be better applied for this project. It worked well and we updated our stage/gate process accordingly.

The third case was fundamentally different and brought with it some demanding challenges in technical, communication, and people management areas. This case study

is known as Global Wi-Fi Rollout (still ongoing). It was our chance to test the Innovation Scorecard in a non-agile software development area.

Based on the positive feedback received from the Red Hat teams involved in this initiative, we can report that the Innovation Scorecard has been well received and has made significant contributions toward identifying and improving work processes including communications, empowering people, and reducing error/bug rates.

### We successfully completed the following activities:

- Development, design, and verification of an Innovation Scorecard Methodology (certified by the Czech Society for Quality)
- Publication of our Innovation Scorecard Methodology by Springer Verlag (Germany) at the end of 2021
- Series of related research articles published by the Project Management Institute (PMI) during 2021
- [Innovation Scorecard Presentation](#) in September 2021 (organized by Red Hat) who wish to apply the Innovation Scorecard in their work area
- Made [associated templates and worksheets](#) freely available on our website ([iscorecard.org](https://iscorecard.org)) to provide help and support to those who wish to apply the Innovation Scorecard in their work area


## PROJECT: Vega Project

### ACADEMIC INVESTIGATORS:

Gabriel Szász (Masaryk)

**RED HAT INVESTIGATORS:** Nikolaos Moraitis, Zdeněk Švécar, and Filip Hubík

Our research group has already collected some intriguing evidence that rotation has a significant impact on the parameters and evolution of stars. We just need to prove these findings on a larger scale to convince a broad scientific community. In order to build the new model atmosphere grid for the rotating main-sequence stars, which will help prove our case, we needed more computing power.

The project is using Red Hat® OpenShift® as a platform for execution and monitoring of scientific computations. An OpenShift based framework supports scalable parallel open hybrid computing and enables easier distribution, scalability, and flexibility of already written code that cannot be compiled on a supercomputer or parallelized otherwise. Through a Red Hat Research donation to Masaryk University—Faculty of Science, we received sixteen servers. We delivered supporting IT infrastructure (e.g., rack, PDUs, etc.), and the stack is now built in the Masaryk University datacenter. At the moment, we are in the process of streamlining the OpenShift installation onto sixteen bare metal nodes. This will be our basis for actual modelling of stellar atmospheres. 





# AI ON INTEL®



**NOW BUILD THE AI YOU WANT  
ON THE CPU YOU KNOW.**

**Learn more at [ai.intel.com](https://ai.intel.com)**