RH RQ

Bringing great research ideas into open source communities

Daniel Gruss and Martin Schwarzl

When is it secure enough? Vulnerability research and the future of vulnerability management

+

Meet osnoise

Preserving privacy in the cloud

Open source research opportunities abroad

Red Hat Research Quarterly
Volume 4:3 | November 2022 | ISSN 2691-5251

Years to build the team.
Months to build the app.
One moment to see them launch.
This is what connecting your clouds feels like.
Red Hat
redhat.com/ourcode

RESEARCH
QUARTERLY

VOLUME 4:3

Table of Contents

Departments

Features

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

RESEARCH QUARTERLY

VOLUME 4:3

From the director

To be more secure, we need to get more open

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.

Suppose I came to you and told you that your front door had a cheap deadbolt that could be opened with a credit card, and that lots of people who want to rob you would soon know about it. You'd probably thank me very much for the information as you left for the hardware store to buy a better lock.

On the other hand, suppose I told you that deadbolts aren't all that helpful anyway, because people who really want to get into a house can break a window or kick in the door. You'd probably say, well, that's a risk I can live with, and tell me not to be such a worrier.

We make this same trade-off every day in computer security, and understanding the nuances around it—which vulnerabilities are really worth fixing, which ones aren't, how do we assess the difference—is a huge part of the work Daniel Gruss does every day. Gruss, the first researcher to appear twice on the cover of this magazine, spends his days as a security researcher at TU Graz in Austria trying to find security holes and figure out how to plug them, if they are worth plugging. He and his student Martin Schwarzl talk with Red Hat Vice President of Product Security Vincent Danen in this issue about the trade-offs they have to make when deciding where to spend their time as researchers. They also discuss the similar tradeoffs hardware and software makers must make to deliver products that are secure enough, but not so secure that you can't use them for anything. I finished the interview more convinced than ever that open source is a vital part of mitigating these problems, and I'm encouraged to see hardware and software makers moving in that direction too.

Encryption is a big part of making things secure, of course. This issue features an in-depth look at Fully Homomorphic Encryption: the set of techniques one can use to encrypt information such that it can still be computed on, without the party doing the computation being able to interpret the results. This sounds entirely impossible to me, but in fact with sufficient computing time it is possible to do things like addition and multiplication on encrypted data without knowing the decrypted inputs or outputs of the computation. Until recently, this kind of computing has been so time consuming that it is impractical for any real-world use. Researchers at Boston University, however, have found that using an FPGA programmed specifically for these computations can make the work so much more efficient that it might, with certain caveats, become practical in the real world. Why would you want to do this? Read Red Hatter Lily Sturmann and BU PhD candidate Rashmi Agrawal's piece on the technique to find out.

Finally, I am very happy to share that we are featuring another piece by Linux kernel developer Daniel Bristot de Oliveira in this issue, this one on ways of identifying and reducing noise in kernel processing to maintain real-time consistency guarantees. Daniel has a special talent for describing very advanced techniques in a way that even non-kernel-developers like me can understand, and I'm delighted to have his continuing contributions to the magazine as the Linux kernel expands into the exotic domains where real-time safety must be guaranteed. Daniel's work exemplifies what happens when open source research meets open source practice, and I'm proud that RHRQ can showcase that connection.

DevConf.US at the edge

This year's conference showcased the many flavors and functions of edge computing.

by Gordon Haff

Now in its fifth year, DevConf.US was back in person at Boston University this past August. Aimed at community and professional contributors to free and open source technologies, DevConf.US included talks and plenty of informal discussions about the usual wide range of topics, including containers, serverless, GitOps, open hardware, growing open source communities, supply chain security, and more. However, one of the most discussed topics—especially in Red Hat CTO Chris Wright's keynote—was edge computing.

As Chris put it, edge is a "big distributed computing problem that's about bringing compute closer to producers and consumers of data." It's driven mainly by the massive amounts of data collected from sensors and the desire to process and act on that data—much of which can't be sent back to a cloud for cost, bandwidth, or latency reasons. (There are also often restrictions on where data can be stored for regulatory reasons.)

The range of edge talks made clear that edge isn't a singular thing. To a retail chain, edge may mean what used to be called remote-office/branch-office (ROBO). To a telco, a platform for software-defined networking. To an automaker, computers in a car. To a manufacturer, industrial control systems.

Like any large distributed system, edge architectures can get complicated. Red Hat's Ishu Verma and Chronosphere's Eric Schabell introduced portfolio architectures, which provide a common repeatable process, visual language and toolset, presentations, and architecture diagrams for edge and other complex deployments. They document successful use cases deployed at multiple customer sites using a variety of open source technologies.

IT is "swinging towards decentralization," said Wright.

About the Author
Gordon Haff
is a Technology Advocate 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.

Today, we're swinging towards more decentralization.

The architectures are themselves open source, and they document, for example, a 5G Core deployment with a logical view and a schema or physical view—along with a more detailed look at the individual nodes or services.

One common theme in many edge discussions is the need to work with resource-constrained hardware, given that edge installations often have quite limited cost, power, and space budgets. They may also have to deal with intermittent network connectivity. Red Hat's Hugo Guerrero discussed how frameworks and runtimes like Quarkus allow existing cloud developers to reuse their Java knowledge to produce native binaries that can run with resource-constrained devices. He also covered how event-driven architecture helps applications continue running locally and then synchronize data when connectivity is available.

Another example of working with small devices, such as a Raspberry Pi, was covered by Red Hat's Jordi Gil in his talk about Project Flotta, which provides edge device management for Kubernetes workloads. Containers on the edge was a popular topic and an area of active engineering investigation. For example, there are slimmed-down versions of Kubernetes for specific edge use cases, and further work continues. Multiple variants are needed because different use cases have different assumptions and requirements around factors like available resources, network reliability, and high availability.

But is Kubernetes, which even in slimmed-down form consumes a certain amount of resources, even needed? Both Sally O'Malley and Dan Walsh of Red Hat provided a look at some of the current explorations into managing edge workloads in the most constrained environments using Podman without Kubernetes.

Managing edge devices is not an easy problem. Dan raised multiple questions that have to be answered: How do I update hundreds of thousands of nodes? What happens if the update fails? How do I update all of the applications on these nodes? How do I add and remove applications to these nodes after they have been distributed? How do I make sure that these computers are safe and secure? He also described some approaches his team is working on to find answers to these management challenges without compromising the security of the edge devices.

Sally described one of these approaches in her talk. FetchIt is a tool for remotely managing workloads with Git and Podman but without requiring Kubernetes. Podman provides a socket to deploy, stop, and remove containers. This socket can be enabled for regular users without needing privilege escalation. Combining Git, Podman, and systemd, FetchIt offers a solution for remotely managing machines and automatically updating systems and applications.

Edge technology reflects the IT industry's long-term pendulum swing from centralization to decentralization and back again. And, as Chris Wright put it, "Today, we're swinging towards more decentralization." RH RQ

Research Day Europe highlights innovation and Czech hospitality

by Matej Hrušovský

Red Hat Research Day Europe 2022 in Brno took place on September 15. The event brought together people from industry and academia, customers and partners, government institutions, and educational organizations. Attendees heard from leaders in open source innovation about their latest work and the future of open source research.

We conceived Research Day, first held at the Red Hat Summit in 2019 in Boston, as a way to promote information exchange and idea sharing among university researchers, Red Hat engineers, and open source users in the form of Red Hat partners and customers. Part of the mission of Red Hat Research is to support work that is at once cutting edge and grounded in the practical needs of open source projects and the companies and institutions that use those projects every day. We walk a fine line balancing these two contradictory concerns, and the more information we have about what motivates users of open source software, the easier it is to understand what areas are ripe for improvement versus which ones should be left alone.

Many of the projects conceived or just launching at that first event are now reaching maturity, as you can see if you visit our latest set of virtual posters on research.redhat.com. Three years later, this latest in-person event in Brno was both a welcome return to our original format and a new departure in that we did not attach it to a major Red Hat event.

Red Hat Research

Red Hat Research Day
by the numbers

100+ guests

11 speakers

9 research talks

7 countries

4 networking sessions

1 party with traditional Czech food, drinks, and music!

About the Author
Matej Hrušovský has been with Red Hat for more than nine years, seven 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.

The mission of Red Hat Research is to support work that is at once cutting edge and grounded in the practical needs of open source projects.

Meeting face to face is a highlight of any research gathering, even more so when we've been waiting so long.

RESEARCH DAY SPEAKERS

Speakers have made their slides available on the Red Hat Research website. To see individual recordings of all Research Day talks, go to the Research Days 2022 playlist on YouTube.

Using static analysis for microservices

Tomáš Černý, Baylor University

Static analysis in the context of microservices has to contend with decentralization and heterogeneity across codebases—something conventional tools fail to address. This talk introduces static analysis prototypes adapted to microservice development practice to perform multiple tasks.

Analyzing the security certifications landscape: does certification help security?

Ján Jančár, Masaryk University

Security certificate documentation such as Common Criteria or FIPS 140 contains a wealth of information for vulnerability assessment and security research. However, because the documents are not standardized or machine readable, mining the documents for information is difficult. This talk proposes an automated way of gathering this information and making it more readily usable for analysis.

LEGO for 6G: a modular open access end-to-end network architecture to enable B5G/6G research

Abhimanyu Gosain, Northeastern University

This talk provides an overview of globally identified 6G candidate technology areas and suggests a roadmap for leveraging research testbeds to influence the 6G vision and accelerate the 6G research and development lifecycle. It includes the presentation of an end-to-end programmable virtualized cloud native B5G architecture built on Red Hat OpenShift.

Building a science lab with ContainerSSH

Nikos Tsipinakis, CERN; Janos Bonic, Red Hat; Sanja Bonic, Red Hat

ContainerSSH simplifies the provision of SSH in a multiuser environment, using webhooks or Kerberos to provide access independently from the underlying operating system. It is currently being tested as a pilot for the LxPlus Service at CERN, which provides access to roughly 2,000 concurrent users.

Unleashing affordable cloud resources with SpotOS

Assaf Schuster, Technion, Israel Institute of Technology

SpotOS addresses the high price of renting cloud resources by facilitating the rental of spot instances. Using optimization algorithms, predictive machine learning, efficient external storage, and stateful task management, it enables more efficient resource usage at a lower cost. This talk surveys both the challenges and the opportunities in SpotOS.

Lightweight always-on network latency monitoring with eBPF

Simon Sundberg, Karlstad University

Current software-based passive network monitoring tools often struggle to keep up with the traffic as packet rates increase. Can emerging eBPF technology enable lightweight passive network latency monitoring by implementing evolved Passive Ping (ePPing)? ePPing can run on Linux devices at speeds of over 10 Gigabits per second on a single core and has a considerable impact on the users' quality of experience.

Breaks between sessions and a great location provided a natural opportunity for networking.

All speakers received a universal programmable board called Logic as a gift, and three attendees won Logic in a raffle. Logic was created by Robotárna, a local educational organization.

Programming languages to runtime execution: a continuum for service orchestration in dynamic edge/fog environments

Frédéric Le Mouël, University of Lyon, INSA Lyon

Data-heavy applications need low latency, geo-data storing and processing, and privacy and security protection, requiring new hybrid public and private edge/fog/cloud architectures. This talk presents ways to rapidly program and prototype FaaS-based distributed dataflow applications, allowing easy redeployment according to dynamic change in the environment and preventing single points of failure.

Red Hatters Nora Haxhidautiova and Martin Ukrop were among the Brno-based planners of Research Day.

These are just some of the interns whose help with the event was absolutely invaluable.

Testing the reliability of systems with unstable or low-quality network connectivity
Miroslav Bureš,
Czech Technical University

This presentation introduces a specialized technique to test the processes of a system in situations when parts of these processes are affected by limited or disrupted network connectivity. The technique is explained in a practical example: a sensor network system for combat casualty care developed by the Czech Technical University in Prague in cooperation with other Czech partners.

Parametric log checking
Aleš Smrčka, Brno University of Technology

One of the main problems of log analysis is the interleaving of log messages related to different sessions, resources, and purpose-specific objects; logs are parametric. This talk describes a way to monitor a system and determine whether it is working correctly, based on the system's log messages and the provided specification, with examples of a prototype monitoring tool applied to several different systems.

A SUCCESSFUL RETURN

Networking is one of the best outcomes of these in-person gatherings, and Research Day 2022 was no different. The event fostered many conversations between engineers, customers, partners, local management, academics, and researchers, and the foundations were laid for potential new projects and partnerships.

Research Day 2022 was also the grounds for the first run of the new Brno Student Ambassador Program, aimed at involving local interns as guides for international speakers. Fifteen interns took part in this initiative, and the feedback for them (and from them!) and for the program was overwhelmingly positive.

It's been a while since Red Hat Research has been able to hold an in-person event—the last one took place in Brno in January 2020. Everyone knows what happened next: a worldwide pandemic. We were all very excited in early 2022 when our team decided to organize the next research day. Judging by the feedback received internally and externally, it was a success!

It was exhilarating to see how lively such an event can be and what value and excitement it can bring to both speakers and attendees. If we can help it, the next one will come much sooner.

NVIDIA

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
Learn More About **DGX on OpenShift** @ nvda.ws/dgx-openshift

When is it secure enough?

Vulnerability research and the future of vulnerability management

An interview with **Daniel Gruss** and **Martin Schwarzl**

Interview

Security researcher and professor Daniel Gruss is an internationally known authority on security vulnerabilities. Among the exploits he's discovered with his research team are the Meltdown and Spectre bugs, and their software patch for Meltdown is now integrated into every operating system. Frequent collaborator Martin Schwarzl, a PhD student in Daniel's CoreSec group at Graz University of Technology (Austria), joined Daniel for an interview with Red Hat Vice President of Product Security Vincent Danen.

Vincent Danen: A lot of your research is on security-related topics. What got you interested in security in the first place, and why are you teaching it?

Daniel Gruss: At first, I was more interested in operating systems, and I wanted to continue teaching. I asked my professor, "Can I continue teaching?" He said, "No, that's not possible. If you finish your master's, you'll go out into industry." I said, "But there are these older people here, not much older than me. I don't know what they are doing, but they still teach classes," and he said, "Oh, you mean the PhD students?" And I was like, "Yeah, I think I could do that."

He said that was the worst motivation to start a PhD he'd ever heard! But it worked out, and he became my PhD supervisor. That's how I ended up teaching, and it's also how I got into security, because that was the topic of the research group here. A coincidence, but I think it's a very natural fit. If you have a background in operating systems, you have an easy time moving into systems security.

Vincent Danen: I agree with that a hundred percent. I built my own Linux distribution, and it teaches you a ton. I'd never dug into kernels before—that's an eye opener! If you can do that, you can do anything. It really gives you a good understanding of the security side.

Daniel Gruss: Martin actually had a clearer vision that he wanted to go into security.

Martin Schwarzl: I attended a higher technical college with a focus on computer science, and I had a good teacher who was strongly encouraging students into security. We did a lot of Capture the Flag (CTF) playing and basic web stuff, and from there I fell into this rabbit hole where you get deeper and deeper and want to understand more and more things.

Vincent Danen: A lot of your work is focused around hardware. Intel SGX work, AMD prefetch attacks, register abuses, memory deduplication attacks—all of it's very hardware focused, which is different from a lot of other security researchers. What is it about hardware that interests you?

Daniel Gruss: I think the interesting part is the intersection. There are hardware-only security groups and software security groups, and our group is in the middle. For instance, if you take any application, there are multiple layers involved: You have timing differences caused by a hardware interrupt and also the operating system handling it. So you're observing both of them. The general concept of "application" is implemented on the operating-system level. So you're effectively combining different things, different layers, and suddenly you need some knowledge on all of those parts.

Martin Schwarzl: You also need to find applications that give you all the ingredients to perform these kinds of attacks. You start from the high-level application down to the operating system, down to the hardware. That makes it super interesting.

About the Author
Vincent Danen
lives in Canada and is the Vice President of Product Security at Red Hat. He joined Red Hat in 2009 and has been working in the security field, specifically around Linux, operating security, and vulnerability management, for over 20 years.

Vincent Danen: It makes it really random too, right? Because you have the differences in timing, whether it's ARM architecture or the differences between Intel and AMD. Then you start throwing different hardware at it. Even if it's the same version of the Linux kernel, maybe you're throwing different processors in the mix. That adds a lot of interest to the problem, because now you're looking at it from a whole bunch of different angles, but there are a lot more puzzles to solve.

> I think the interesting part is the intersection between hardware and software…

Daniel Gruss: We've seen that, for instance, with the prefetch attacks on Intel and AMD. We published research on the attacks on Intel in 2016, and we published on AMD last year. There were five years between when we figured out prefetch attacks on Intel and when we successfully tried it on AMD! In that case, it turned out that the timings were inverted. The fastest case on Intel is actually the slowest case on AMD. Very curious, and not what we would've expected.

Vincent Danen: It makes you question every single assumption that you have.

Daniel Gruss: Yes! I also like the insights we gained from the Platypus paper about this ability to perform a power side channel attack from software. Because power side channel attacks are super powerful—in theory, there's nothing you can do to defend against them. Even if you try something complex, like masking on the hardware—which is super popular—even if you can perform a super fine-grained measurement on a specific point, you can still break up those schemes. That's why people say, “Okay, let's not just mask it one time, let's split it up further.” The idea is that if we split it up often enough, an attack takes so much time, and it's so expensive, that it's not worth it.

But there's no ultimate security, which makes it difficult sometimes to determine whether what you've done is good enough. And it's difficult to convince other people that what you're doing is good enough, because there's never 100% security.

Vincent Danen: I wish more people understood that. Because you get a lot of people who assume, “Oh, there's a vulnerability. We have to fix it.” But sometimes the benefit you get from exploiting that vulnerability is so small, while the expense to fix it is so high, that you just think, why are we worried about this? It makes me think of some of the Spectre and Meltdown stuff and some of the mitigations to prevent it, for example, turning off hyperthreading.

Daniel Gruss: Yes, but that's a bit dangerous too. We've seen a similar effect with the COVID situation, where everybody got the vaccine then said, “Oh, COVID wasn't so bad. We could have skipped all of this.” The question is, was it not so bad because of the vaccine? Or was it not so bad regardless of the vaccine? Likewise, we don't know what would have happened if we hadn't had the Meltdown and Spectre patches. For Meltdown, I'm absolutely sure we would've seen exploits in the wild. It was just not interesting anymore because the systems were all patched rather quickly. For Spectre, it's super tricky. Spectre is far more tricky to exploit than Meltdown.

Martin Schwarzl: In some cases, especially when it's inside the same process, it's difficult to fix these optimizations because you lose plenty of performance, and that's the opposite of what you usually want.

Vincent Danen: That's really important—there's that middle ground you want to hit. You don't want to go to either extreme, either: “Don't bother patching it because performance is king and that's all I care about,” or, on the other side, “I don't care about performance. I want perfect security,” which, as you say, doesn't exist, and you're going to lose all of the performance for the sake of a false sense of security.

The middle ground is somewhere in between, where you can get a decent amount of performance and a decent amount of security.

It all depends on your risk appetite and what it is that you're willing to sacrifice.

Daniel Gruss: I think this direction will be even more relevant in the near future, because we are running into a global climate and energy crisis. We have exponential growth in energy consumption, and at some point people will start asking, “Why do we run this security feature again? Can't we just turn it off?” Why do we have these patches installed? How much do they cost? We don't know how much they cost. We are not measuring it.

Vincent Danen: That is an interesting point. If you do all of the security mitigations and you're increasing power consumption, what does that do for the overall global footprint of that cost? When the benefit of what you're defending against is not worth that expense…

Daniel Gruss: Exactly.

Vincent Danen: … then we have to revisit our assumptions.

Daniel Gruss: Yes. That's absolutely the direction we will pursue in the future: policies with a more fine-grained selection for which workload gets which security level and which types of protection, and do that on a workload basis, on a computer basis, on a user basis. Do I share the cache? Do I not share the cache?

Vincent Danen: That really speaks to knowing your environment, right? You don't need to put in the same security mitigation in a development environment versus a production environment. And even in a production environment, it depends on the type of data you're dealing with.

Daniel Gruss: Right—and also on the exposure of the system. Is it possible for an attacker to run a binary on that system? If that's not possible, I can rule out a lot of attacks already.

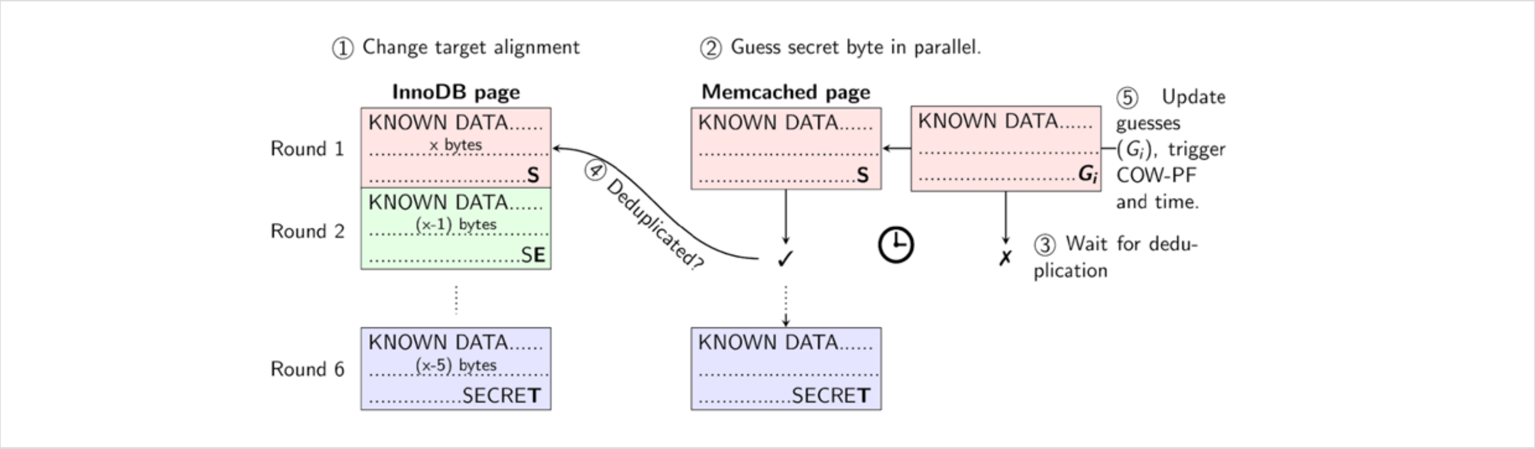
Vincent Danen: It really requires the end user to understand what they're doing with their devices. Right now, I don't feel like most people do. They just assume they're running a computer, and since they care about security, they have to have all the security, patches, mitigations, and so on in place, but they don't know if those things are useful or valuable. So I like this kind of capabilities view, because I think it's going to become really important.

Another question I had was about the complexity of the research you're doing. A lot of it is very complicated, and some of it is probably very difficult to prove. You can theorize it, and then you have to go out to find it. How long does it typically take to theorize and then test for those vulnerabilities?

Daniel Gruss: I would say that we build up a lot of theories and like 90% or maybe 95% of them just don't work. We try them, and then we realize, “Oh, it doesn't behave that way.” The remaining maybe 5% that we pursue, some of them turn out to work in the direction we thought they would, but they're not exploitable, so they're not interesting for our purposes. We learn something about the system, but nothing we want to investigate further or publish about.

And then there's a very small percentage left over where we can say, “Oh, that's, that's so interesting!” We can evaluate the security implications of this effect and publicly document it.

> But there's no ultimate security, which makes it difficult sometimes to determine whether what you've done is good enough.



High-level conception of the InnoDB Reorganization attack. By exploiting the reorganization of data in InnoDB, the attacker can leak secret database data byte by byte.

Vincent Danen: When 95% don't work, how much time do you spend before you decide you've reached the point of diminishing returns?

Daniel Gruss: That's an excellent question. There is no algorithm you can follow to tell you to spend one more week on something. There's nothing except your intuition, your gut feeling basically, which is tricky.

It also changes over time. When I was getting my PhD, I was rather efficient. When I run experiments now, I'm much slower, because I have so many more concerns: "Oh, but what if this goes wrong? And what if this goes wrong? And have you considered this?" There are many things to think about, and you can lose a lot of time even before you start experimenting. It's not so much about, "Should I invest one more week?" It's that you bring your big backpack of thoughts. So you lose efficiency over time, but you also build up more and more knowledge, which helps your intuition on whether you should invest more time in something.

Vincent Danen: That's the importance of experience. I was going to ask Martin about this, too, since you are earlier in your career. From the perspective of the teacher, maybe there's more caution, and then there's the student who's doing the research, who is—I don't want to say reckless, but a little bit more…

Daniel Gruss: … let's say, optimistic?

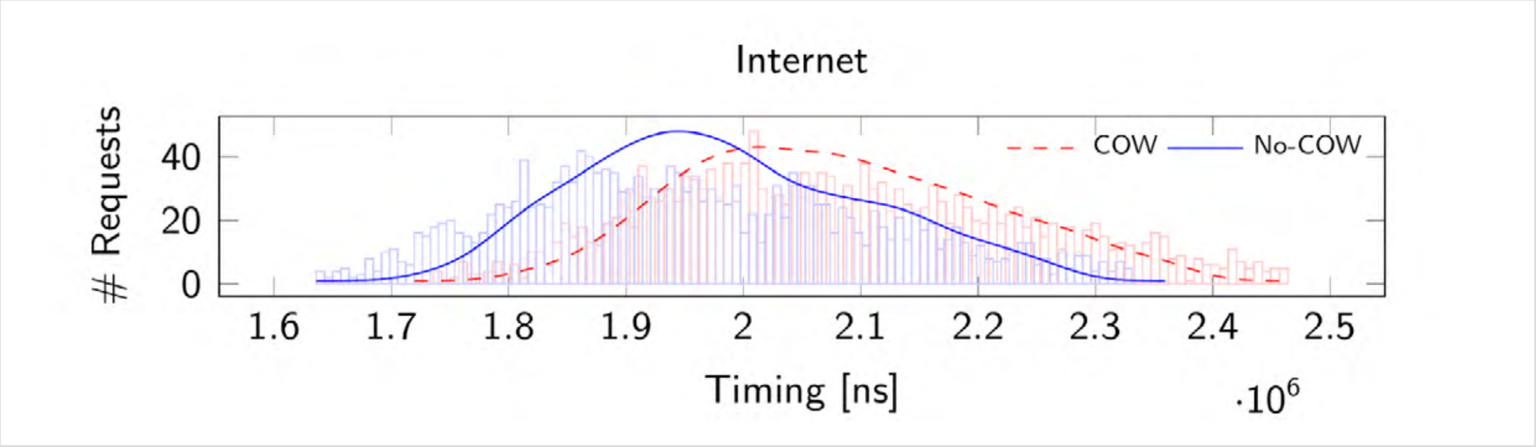
Martin Schwarzl: That depends on the topic. How well started is something you are trying to do new research on? If you have something like cache attacks, where there is 15 years worth of research already, and you want to build upon it, you can count on certain things. But if you go into a completely new direction that no one has any idea about, it takes a few months to get in touch with the topics.

And another piece is, of course, luck on the hardware you're running the stuff on.

Daniel Gruss: For example, there was an article by Anders Fogh that basically described Meltdown, but it didn't work. The Meltdown attack that works is essentially the same. It's on a different machine, and the code we had looked a bit different, but the idea is the same, and it works.

Vincent Danen: Was it due to access or lack of access to the particular hardware?

Daniel Gruss: I worked with Anders before on the prefetch side channel attack, and the laptop he most likely tried it on was one he also used during the prefetch side channel attack paper. I suspect that the laptop just has a very, very short



Histogram of network timings of a web server running memcached using a remote memory-deduplication attack with a 16-page amplification factor.

transient execution window. And that means you don't have much time to read the data, encode it, and then leak it on the other side. So yes, hardware makes a big difference. One person in our group always had luck with his laptops. We had so many attacks that we tried on different laptops, and usually if he'd started something, you'd hear, "Oh, it worked!" Then someone else tries on their laptop, and nothing works.

Vincent Danen: It's almost infuriating. That dependency is really hardware based, not just on the chip, not just the CPU, but all of the other pieces and combinations of pieces.

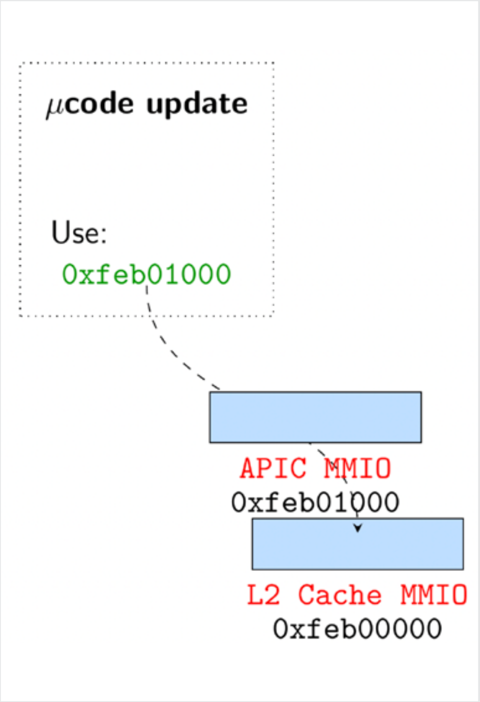
Daniel Gruss: Yes! But then that's the next step that we take. We need to figure out why it doesn't work on other machines and how we can make it work.

Martin Schwarzl: Can we talk about the ÆPIC leak? So we were doing research in a completely different direction. We were trying to understand the Intel microcode on these Goldmont devices.

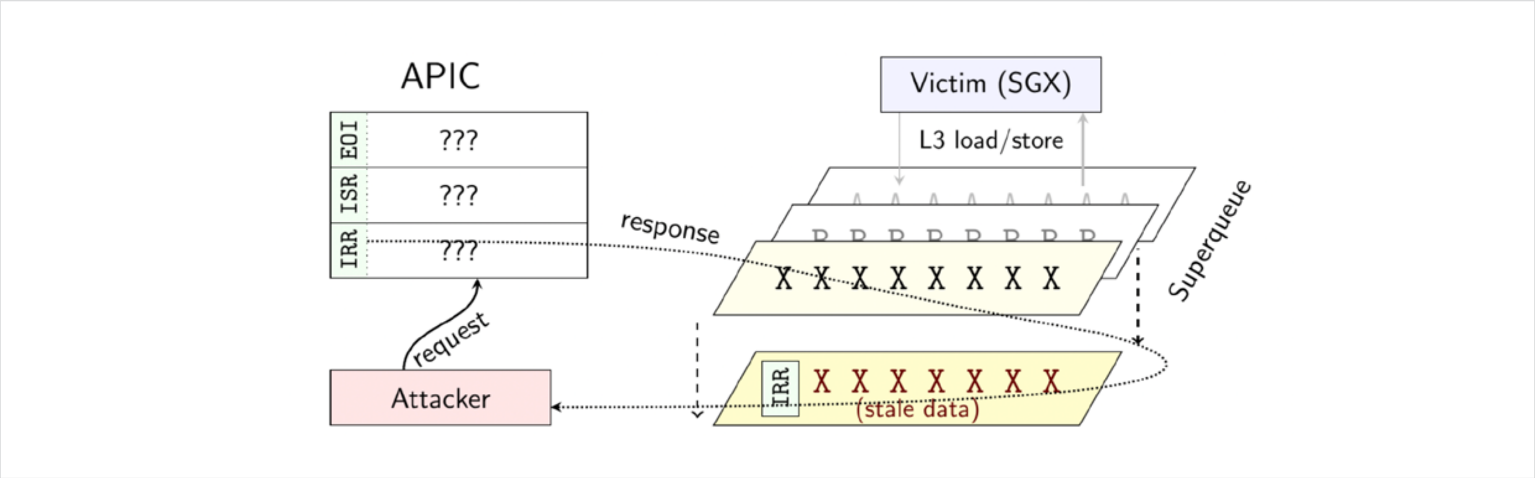
Daniel Gruss: I was very skeptical. I didn't think there was anything in that direction that would be publishable.

Martin Schwarzl: And then we somehow had the idea to use the memory sinkhole and read from the APIC to sample data during a microcode output. During the reverse engineering, we saw in the decompiled microcode that a special physical page is used during the update routine. We wanted to read out the plain microcode update, but it still did not work. Then my colleague Pietro did that on every single machine in our lab. There was exactly one device where we actually read some data, and it was this notebook we got as a gift from Intel.

From there, we started doing more research and found the ÆPIC leak.



Failed exploit attempt to leak the μcode update. By leveraging the memory sinkhole, an attacker tries to leak the full microcode during an update.



ÆPIC LEAK: On most 10th-, 11th-, and 12th-generation Intel CPUs, the APIC MMIO undefined range incorrectly returns stale data from the cache hierarchy. This attack can be used to extract arbitrary data from SGX enclaves, such as cryptographic keys.

It was completely lucky that we had the device there, because if we didn't, we might have decided it's just not exploitable. We tried 15 different devices that did not work, but on the sixteenth one it did.

Vincent Danen: So there's luck, but there's also tenacity. And in your case, there's the availability of hardware.

Martin Schwarzl: Also, you need this drive to always look for the needle in the haystack. If you don't have the stamina, you will probably stop after one or two days.

Daniel Gruss: People often start a topic and then, after a few days, say, "Ah, there's nothing there." But did you try this? No. Or this? No.

Vincent Danen: You have to exhaust all the possibilities, right? One of the other things I wanted to ask you about was your involvement in patching and fixing vulnerabilities. A lot of times, finders of vulnerabilities are just there to break things, not necessarily to fix things. But you take this other approach, where you are also instrumental in fixing things, due to your understanding and the research that you've done.

How receptive are vendors to that? When you reach out and say, "Hey, we found something," what does that look like?

Daniel Gruss: So there are researchers from industry and researchers from academia, and there may be differences. In academia, it's now expected that you propose some defenses or mitigations in your publication. That's why we start thinking about defenses while working on these publications. Sometimes if an idea is strong enough to be a publication by itself, we also publish it separately. And we constantly try to stay in touch with the vendors. With all of these microarchitecture vulnerabilities, that's primarily the CPU vendors, but often enough also software.

In general, they really like to hear from us. They don't necessarily want to share ahead of time what they are planning to do, but even that sometimes works. We had some very successful collaborations with Red Hat, where we worked on mitigations together. But, for instance, on the Intel side, we sometimes got microcode before it was released. Not in all cases—for instance, for ÆPIC, we had to wait until everyone got it. Of course, it would be interesting, because then we can look at it earlier and see whether it works. There have been cases where we couldn't tell them in time. If we get the microcode patch and the disclosure is one week later, we need some time to test it.

Vincent Danen: Given how expensive it is to release a patch, particularly when it comes to hardware, you'd think they'd want you to look at it. Because that's not an easy thing to deploy—all these different microcode and firmware updates and whatnot. If you have somebody willing to test this on your behalf for nothing, other than just time and accessibility, you would think, "Hey, these guys broke it. Let's give them something and see if they can still break it or if this fix actually works as we think it will."

Daniel Gruss: Yes, although the "for free" part is a bit tricky. The researchers are not free on my side! Intel has funded my research group in the past, and I'm very happy that we also have this collaboration with Red Hat. That currently helps me to stay above water.

Vincent Danen: That's a really good point. So Daniel, you're a researcher, you're a teacher, and in the security field, in particular, we hear a lot about burnout and rising rates of burnout for security professionals. Here you are, a security professional and a security teacher, so given that risk of burnout, how do you deal with work-life balance? And how do you set an example of work-life balance for your students? Because I think that's part of teaching as well.

Daniel Gruss: That's tricky. If the funding situation is good—and it has been good in my group for some time—that helps. If you have projects where the project work is for publications or working on mitigations that you can then publish, that's perfect.

When the funding situation is more difficult, you have to pick up more of those projects where you have to invest a lot of time on deliverables. And for the PhD students, that is a time where you're also studying, and there's a bit of variance in how intensive this is for you, how motivating, or how frustrating.

> The middle ground is somewhere in between, where you can get a decent amount of performance and a decent amount of security.

For me, the PhD, although I was working a lot, was the best time of my life. Even though I had a lot of teaching and a lot of other stuff, it was what I wanted to do. I picked my own research topics. I could focus on that research. I could publish it. I had a lot of freedom, and I enjoyed that freedom. It depends on how much energy you can draw out of that. Right now, I don't manage to stay within healthy working hours because there's too much project work to do <laughs>.

Vincent Danen: But if you enjoy it, I think there's that passion part. If you're passionate about the work and you get energy from it. I think sometimes that tips the scales a little bit, right?

Martin Schwarzl: Most of the time, you're focusing on a thing you want to do or getting things done so that you can work on new stuff. I think during COVID times, it was demotivating, because usually as a reward for your work you can go to conferences at nice locations and have another one or two weeks of vacation afterward to celebrate or relax after your heavy workload.

That was completely gone, so you just had the standard work, the project work, the teaching, of course, supervising bachelor's and master's students. You're still working the same amount, but the motivation of knowing that once you get this thing accepted you can go somewhere was gone.

Daniel Gruss: Yes. We were at conferences last week, and it was so good to be at conferences again. I hope we get a bit more of that now.

Vincent Danen: Agreed.

On that note, thank you for taking some time from your busy schedule to share your thoughts with us today. This has been awesome.

RESEARCH
QUARTERLY

Red Hat

VOLUME 4:3

Feature

Meet osnoise, a better tool for fine-tuning to reduce operating system noise in the Linux kernel

Research on the root causes of OS noise in high-performance computing environments has produced a tool that can provide more precise information than was previously available.

by Dr. Daniel Bristot de Oliveira

About the Author
Daniel Bristot de Oliveira is a Senior Principal Software Engineer at Red Hat working on developing the real-time features of the Linux kernel. He is an affiliate researcher at the Retis Lab and an active member of the real-time academic community, participating in the technical program committee of several academic conferences.

This article is a summary of the paper "Operating system noise in the Linux kernel," published in IEEE Transactions on Computers, Open Access.

The Linux operating system (OS) has proved to be a viable option for a wide range of very niche applications, despite its general-purpose nature. For example, Linux can be found in the high-performance computing (HPC) domain, running on all the TOP500 supercomputers, as well as the embedded real-time systems domain. These achievements are possible because Linux has such great configuration flexibility.

Linux is central in developing the core services that support modern networking infrastructures and the internet with Network Function Virtualization (NFV) and Software-Defined Networking (SDN). The 5G network stack is built on this paradigm, enabling a new set of services characterized by strict timing requirements, on the order of 10s of microseconds.

To meet these tight timing requirements, hardware and Linux are configured according to standard practices from the HPC and real-time domains. To this end, the hardware is set to achieve the best trade-off between performance and determinism. The OS is usually partitioned into *isolated* and *housekeeping* CPUs. The *housekeeping* CPUs are those where the tasks necessary for regular system usage will run. This includes kernel threads responsible for in-kernel mechanisms, such as RCU callback threads; kernel threads that perform deferred work, such as kworkers; and threads dispatched by daemons and users. The general system's interrupt requests (IRQs) are also routed to housekeeping CPUs. This way, the isolated CPUs are then dedicated to NFV work.

NFV applications run either by being triggered by an interrupt or by polling the network device while waiting for packets, running nonstop. While the first case has been extensively studied (for example, in "Demystifying real-time Linux scheduling latency," RHRQ 3:1), this is not the case for the interference suffered by polling threads.

The paper "Operating system noise in the Linux kernel" contributes to this end by:

1. Proposing a precise definition of the causes of OS noise in Linux from a real-time perspective

2. Presenting a kernel tracer that can measure OS noise using the workload approach, while also providing tracing information essential to pinpoint tasks suffering from OS noise, not only that caused by the OS but also noise from the hardware or the virtualization layer

3. Reporting on empirical measurements of OS noise from different configurations of Linux commonly found in NFV setups, showing how the tool can be used to find the root causes of high-latency spikes, thus enabling finer grained tuning of the system

OPERATING SYSTEM NOISE

OS noise is a well-known problem in the HPC field. Generally, HPC workloads follow the single-program multiple-data (SPMD) model, shown in **Figure 1.** In this model, a system is composed of M processors, and a parallel job consists of one process per processor. All processes are dispatched simultaneously at the beginning of the execution. At the end of execution, the process synchronizes to compose the final work and repeats cyclically. Ideally, the parallel job process should be the only workload assigned to the processor. However, some operating system-specific jobs need to run on all processors for the correct operation of the system, like the periodic scheduler tick, critical kernel threads, or others. In this scenario, each local processor's scheduler decisions significantly impact a parallel job's response time. With this background, we can now define OS noise as *all the time spent by a CPU executing instructions not belonging to a given application task assigned to that CPU while the task is ready to run.*

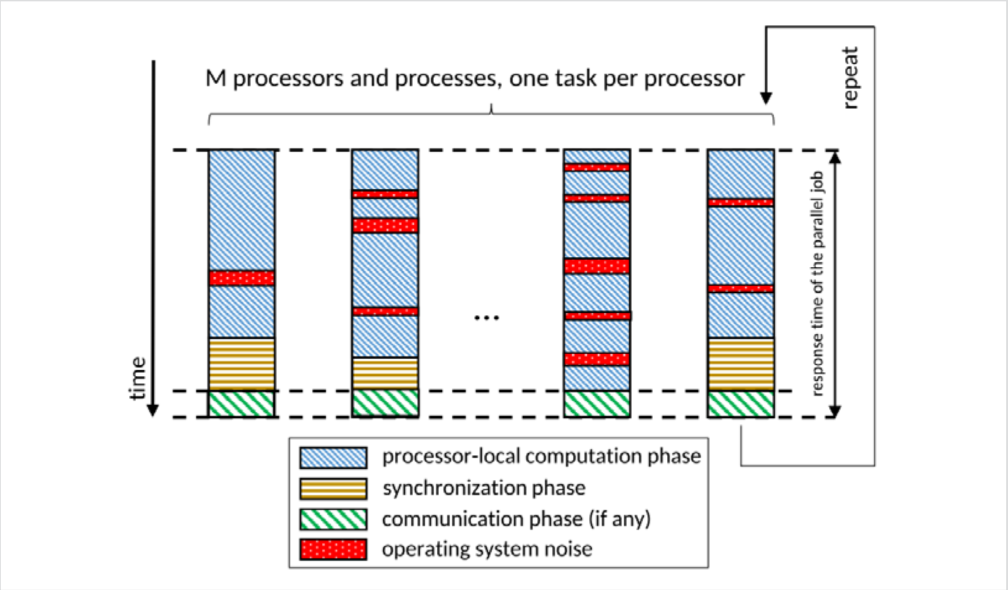

M processors and processes, one task per processor
repeat
time
...
response time of the parallel job
processor-local computation phase
synchronization phase
communication phase (if any)
operating system noise

Figure 1. *The single-program multiple-data (SPMD) model used for HPC workloads and the effects of the OS noise*

In Linux, four primary execution contexts can interfere with a workload: non-maskable interrupts (NMIs), maskable interrupts (IRQs), softirqs (deferred IRQ activities), and threads. Linux's execution contexts are characterized by the following rules (see "A thread model for the real-time Linux kernel," RHRQ 2:3 for more):

- **R1/R2:** The per-CPU NMI preempts IRQs, softirqs, and threads; once started, it runs to completion.
- **R3/R4:** IRQs can preempt softirqs and threads; once started, it is not preempted by another IRQ.
- **R5/R6:** Softirqs can preempt threads; once started, it is not preempted by any other softirq.
- **R7:** Threads cannot preempt the NMI, IRQs, and softirqs.

As the HPC workload runs in the thread context, it can suffer interference from all execution contexts, including other threads.

MEASURING THE OPERATING SYSTEM NOISE IN LINUX

There are two types of tools to measure the operating system noise in Linux: workload based and trace based. Workload-based tools generally run microbenchmarks with a known duration, and they measure the difference between the expected duration of the microbenchmark and the actual time needed to process it. While effective in defining

```
[/sys/kernel/tracing] # cat trace
# tracer: osnoise
#                                _-----=> irqs-off
#                               / _----=> need-resched
#                              | / _---=> hardirq/softirq
#                              || / _--=> preempt-depth                            MAX
#                              || /                                             SINGLE     Interference counters:
#                              ||||               RUNTIME      NOISE   % OF CPU  NOISE    +------------------------------+
#           TASK-PID      CPU# ||||   TIMESTAMP    IN US       IN US  AVAILABLE  IN US     HW    NMI    IRQ    SIRQ THREAD
#              | |         |   ||||      |           |             |    |            |      |      |      |       |      |
           <...>-859     [000] ....    81.637220: 1000000        190  99.98100       9     18      0   1007     18      1
           <...>-860     [001] ....    81.638154: 1000000        656  99.93440      74     23      0   1006     16      3
           <...>-861     [002] ....    81.638193: 1000000       5675  99.43250     202      6      0   1013     25     21
           <...>-862     [003] ....    81.638242: 1000000        125  99.98750      45      1      0   1011     23      0
           <...>-863     [004] ....    81.638260: 1000000       1721  99.82790     168      7      0   1002     49     41
```

Figure 2. *osnoise tracer summary output from ftrace interface*

the amount of operating system noise that a workload might suffer, workload-based tools cannot pinpoint the root causes of the OS noise.

Trace-based methods leverage Linux kernel tracing features to identify the root cause of operating system noise. However, these trace-based methods do not account for how workloads perceive the noise.

Unlike previous work, the osnoise tool proposed in this article takes the best of both workload-based and trace-based methods, pointing to the root causes of the operating system noise while accounting for how the workload perceives the noise.

THE OSNOISE TRACER

The Linux osnoise tracer is controlled via the ftrace interface. The osnoise tracer has two components: the workload component and the tracing component.

The osnoise tracer uses per-CPU kernel threads to simulate an HPC workload. Each thread runs periodically for a predetermined runtime. Each osnoise thread works by reading the time in a loop to detect the time stolen from its execution. A new noise sample is collected when it detects a gap between two consecutive readings higher than a given tolerance threshold. The user can adjust these and other parameters of the osnoise workload via ftrace's interface.

The osnoise tracer leverages the current tracing infrastructure in two ways. It adds probes to existing tracepoints to collect information and a new set of tracepoints with preprocessed information.

At the end of every period, the osnoise workload thread prints a summary containing the amount of noise observed during the current activation, the percentage of CPU time available for the tool, and the maximum single noise observed. The workload threads also print statistics about the noise sources from NMI, IRQs, softirqs, and threads. **Figure 2** shows an example of the trace output.

While Linux has tracepoints that intercept the entry and exit of NMI, IRQs, softirqs, and threads, the manual interpretation of these events is tedious. It requires a large tracing buffer and higher overhead because of the amount of data and might lead to the wrong conclusion if any error is made by the user while analyzing the trace.

The osnoise tracer leverages these tracepoints by attaching a probe to all entry and exit events producing an optimized output. The osnoise tracer adds a single event for each Linux task abstraction, stores the data in the kernel, and creates a single and concise event with all data processed.

Figures 3 and **4** compare regular kernel events and those parsed and generated by the osnoise tracer. The osnoise tracer events have all the necessary information for the direct interpretation of the data while reducing the amount of data written to the trace buffer.

In addition to printing the events reporting the amount of noise generated by each of the Linux

```
    osnoise/3-4417  [003] d.... 203398.433218: sched_switch: prev_comm=osnoise/3 prev_pid=4417 prev_prio=120 prev_state=R+
                                           ==> next_comm=sleep next_pid=5842 next_prio=120
        sleep-5842  [003] d.... 203398.433414: sched_switch: prev_comm=sleep prev_pid=5842 prev_prio=120 prev_state=Z
                                           ==> next_comm=bash next_pid=5802 next_prio=120
         bash-5802  [003] d.... 203398.433830: sched_switch: prev_comm=bash prev_pid=5802 prev_prio=120 prev_state=S
                                           ==> next_comm=bash next_pid=5843 next_prio=120
        sleep-5843  [003] d.h.. 203398.434017: local_timer_entry: vector=236
        sleep-5843  [003] d.h.. 203398.434022: local_timer_exit: vector=236
        sleep-5843  [003] d.... 203398.434629: sched_switch: prev_comm=sleep prev_pid=5843 prev_prio=120 prev_state=S
                                           ==> next_comm=osnoise/3 next_pid=4417 next_prio=120
```

Figure 3. *Example of tracepoints: IRQ and thread context switch events read from ftrace*

```
        sleep-5842  [003] d.... 203398.433413: thread_noise:     sleep:5842 start 203398.433217481 duration 195472 ns
        bash-5802   [003] d.... 203398.433829: thread_noise:     bash:5802 start 203398.433413330 duration 415172 ns
        sleep-5843  [003] d.h.. 203398.434022: irq_noise:        local_timer:236 start 203398.434016335 duration 5627 ns
        sleep-5843  [003] d.... 203398.434629: thread_noise:     sleep:5843 start 203398.433829263 duration 793261 ns
    osnoise/3-4417  [003] ..... 203398.434631: sample_threshold: start 203398.433215747 duration 1414624 ns interference 4
```

Figure 4. *Example of tracepoints: osnoise events read from the ftrace interface with equivalent data highlighted*

task abstractions, the osnoise tracer can also print a trace event anytime a sample read of the clock detects a time gap longer than the minimum threshold. The last line in Figure 4 presents an example of this tracepoint, named **`sample_threshold`**. In the example, the detected osnoise was 1,414,624 nanoseconds long. Because the osnoise workload and tracer are aware of one another, the tracer can precisely define how many operating system interferences occurred between each clock read. This data also reports four interferences in the example, meaning that the four previous events were the root cause of OS noise. Further examples of root cause analysis and details about the internals of the tracer are provided in the full paper and the osnoise tracer Linux kernel documentation.

RTLA OSNOISE

The rtla (real-time Linux analysis) is a toolset provided along with the Linux kernel, and it provides an easy-to-use benchmark-like interface for the osnoise tracer. The rtla documentation can also be found in the Linux kernel documentation.

EXPERIMENTAL RESULTS

This section reports on osnoise usage for the measurement and trace of a system. The system is a workstation with an AMD Ryzen 9 processor, with 12 cores/24 threads. The system is configured with a Fedora Linux 35 server and runs the kernel 5.15 patched with the PREEMPT RT patchset. The rtla osnoise tool collects a summary of the OS noise and a histogram of each noise occurrence.

The system ran with four different setups. The first configuration of the system considered has no tuning applied and is named As-is. The system is said to be tuned when the best practices for CPU isolation are used. In this case, CPUs {0, 1} are housekeeping, and CPUs {2, . . . , 23} are reserved for osnoise workload execution. By default, osnoise workload threads run with the default's task priority (**`SCHED_OTHER`** with 0 nice). However, it is common for NFV users to set a real-time priority for the workload. Additional experiments have been performed

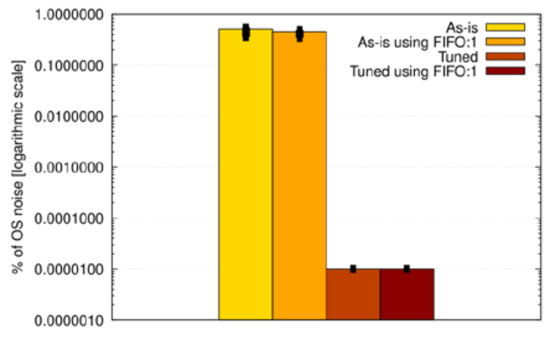
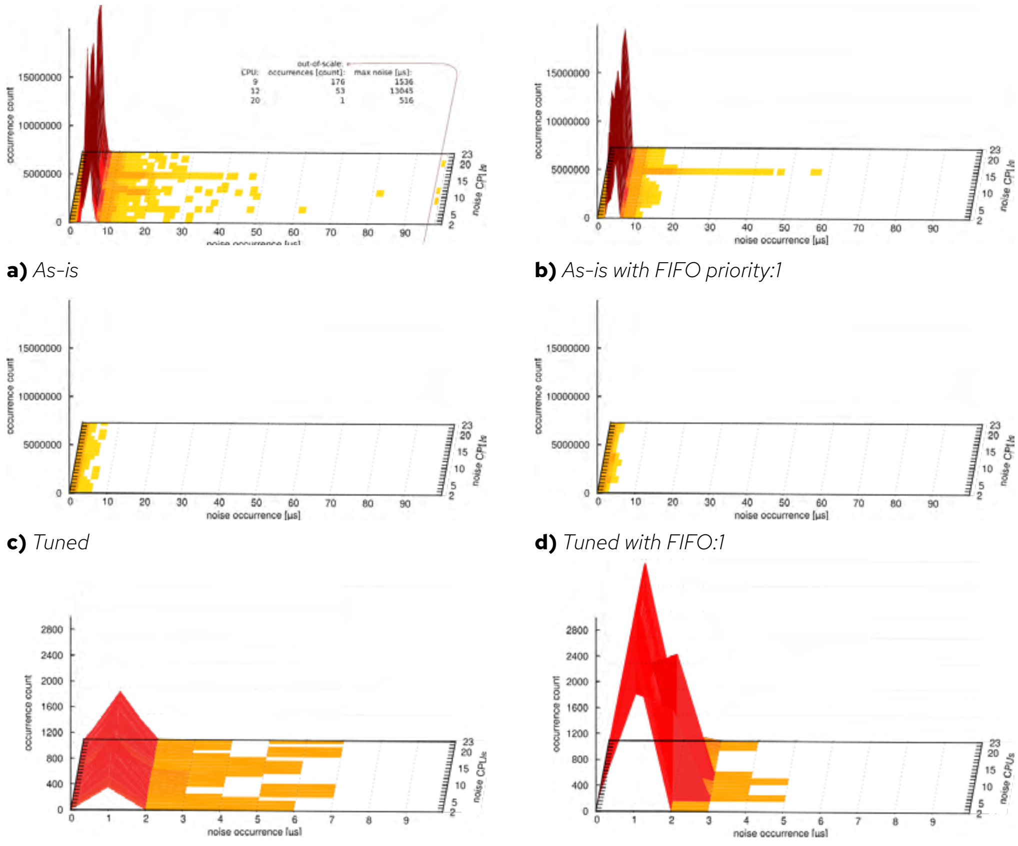


Figure 5. *Average percentage of OS noise observed by the workload on different scenarios. Error bars represent the range between minimum and maximum percentages.*



a) *As-is*

b) *As-is with FIFO priority:1*

c) *Tuned*

d) *Tuned with FIFO:1*

e) *Zoom of Tuned (6.c)*

f) *Zoom of Tuned with FIFO:1 (6.d)*

Figure 6. *osnoise noise occurrence per-CPU histogram under different system setups, mixing CPU isolation tune and real-time priority for the workload (less noise occurrence and less occurrence count is better).*

with osnoise threads running with real-time priority **`SCHED_FIFO:1`** to evaluate this specific scenario. All tests run for six hours.

Percentage of noise

Figure 5 shows the percentage of noise observed on each of the four setups.

The maximum observed total noise was 0.5484% with the system As-is, while the minimum was 0.00001% for both Tuned cases.

OS noise occurrence

An experiment has been conducted for all setups collecting a histogram of each detected noise occurrence. This experiment is important for the NFV use case because a single long noise occurrence might cause the overflow of queues in the network packet processing. The results are presented in **Figure 6**.

With this experiment, it is possible to see the main problem of using the system As-is in **Figure 6a.** The osnoise workload detected 230 out-of-scale noise samples, with the maximum value as long as 13045 μs. **Figure 6b** also shows that using **`FIFO:1`** in the system As-is represents an easy-to-use option to reduce the maximum single noise-occurrence value. Because the workload causes starvation of non-real-time threads, these threads are migrated to the CPUs with time available for them to run.

As-is using **`FIFO:1`**, however, has two significant drawbacks compared to the Tuned options with or without using **`FIFO:1`** in **Figures 6c** and **6d**. The first is the high count of noise occurrences. The Tuned experiment includes the **`nohz_full`** option that reduces the scheduler tick and its consequences. Another difference is the tail latency, which is lower in the Tuned cases. The results with the system Tuned in Figures 6c and 6d show that the tune dramatically changes the entries and duration of each noise occurrence compared to the system As-is. **Figures 6e** and **6f** have been added to visualize the Tuned cases better.

The Tuned kernel was able to deliver consistent results, while the kernel Tuned using **`FIFO:1`** was able to provide below five µs maximum single noise occurrence. That is because the real-time scheduler deferred background OS activities that run as threads. However, the overall noise is higher when using **`FIFO:1`**. The reason behind this is a side effect of the starvation caused by busy-loop tasks running with FIFO priority.

It is worth noticing that these results are only valid for this hardware and setup. Any difference in the kernel and the hardware might change these values, thus the importance of an easy-to-use tool.

REMARKS

The osnoise tool puts together the tracing and the workload, providing precise information at low overhead by processing and exporting only the necessary information for pointing to the root causes of the latency, serving as a good starting point for the investigation.

The experimental results show that the tool can serve as both a tracer and a benchmark tool, facilitated by using the rtla osnoise interface to collect data. It also shows that Linux can deliver extremely low OS noise. But more importantly, the tool can follow the kernel, providing results in the desired scale. Both the osnoise tracer and the rtla osnoise interfaces are an integral part of the Linux kernel, thus accessible to the entire Linux user base.

Feature

Preserving privacy in the cloud: speeding up homomorphic encryption with custom hardware

Fully homomorphic encryption could be a great solution for secure data sharing, if only it weren't so slow. Could an FPGA accelerator be the answer?

by Lily Sturmann and Rashmi Agrawal

Protecting sensitive data from being seen or tampered with, either while it is stored or while it is in transit, has been standard for some time. This practice is especially relevant in cloud computing, which allows access to far more computing power and scalable resources, but involves infrastructure the user does not control, may not be able to access, and therefore may not fully trust.

In recent years, more emphasis has also been placed on protecting data while in use. When this protection includes protection from being seen, it is often referred to as confidential computing. Today, in practice, most confidential computing is done in Trusted Execution Environments (TEEs), in which the CPU enforces the isolation of a region of memory for the sensitive computation. In this memory region only, the data is decrypted for use. Fully Homomorphic Encryption (FHE), by contrast, is a method of confidential computing that goes beyond TEEs and allows for computation on ciphertext while all data remains fully encrypted. FHE has the potential to open a range of new possibilities for confidential computing in medical, financial, marketing, research, and other fields. By enabling computation on encrypted data, FHE encourages people to share their private data without fear of misuse and allows computation outsourcing with fewer security risks.

However, despite its promise, it is not widely used today. There remains a large performance gap between operating on encrypted data and operating on its plaintext form. A calculation that would take one second to perform using plaintext would take more than 11 days to perform using current homomorphic encryption libraries like HElib or PALISADE. This slowdown of about one million times is an unacceptable tradeoff for businesses that would otherwise be interested in FHE and remains a stubborn blocker for practical applications of this discovery.

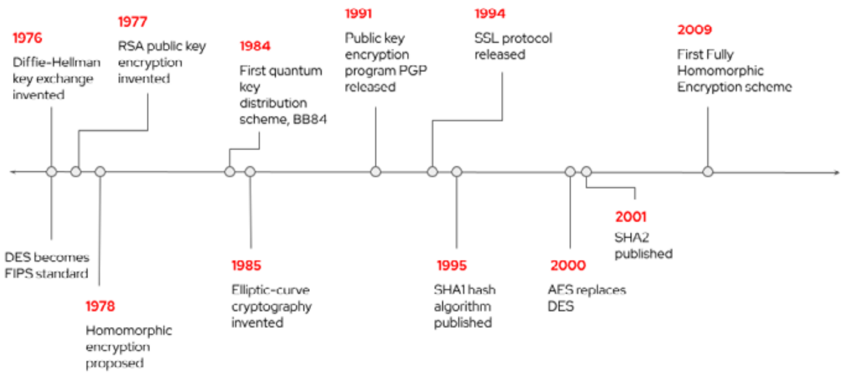
A small research team at Boston University (BU) consisting of graduate student Rashmi Agrawal and Professor Ajay Joshi (along with collaborators from MIT) is working to overcome this limitation and enable privacy-preserving cloud computing using homomorphic encryption. By designing custom hardware accelerators for FHE, they have shown that it is possible to sidestep the biggest obstacle in this field: the performance gap.

AN EXCITING AND CHALLENGING HISTORY

Homomorphic encryption (HE) is a broad term that describes cryptographic advancements to allow computation on encrypted data. The history of homomorphic encryption, of which FHE is the most powerful subset, spans almost a half-century. The idea of HE was proposed in 1978 by Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos of MIT, two of whom are also known as co-inventors of the RSA algorithm. Over the following decades, as new encryption schemes were developed, they were tested for their ability to support HE operations, but they supported them only partially at best. For example, the Paillier cryptosystem supports only additive homomorphism, while Elgamal and RSA encryption schemes support only multiplicative homomorphism. In other words, only addition operations or only multiplication operations can be performed fully encrypted, which limits their real-world applications.

A breakthrough came in 2009 when a graduate student named Craig Gentry described the first feasible construction of an HE scheme that enables both addition and multiplication operations on encrypted data, using lattice-based cryptography. Gentry initially described a scheme limited to evaluating one multiplication and a few addition operations, which would be classified as a Somewhat Homomorphic Encryption (SHE) scheme. This limitation is imposed by the noise in ciphertext generated using a lattice-based encryption scheme. The noise within the ciphertext grows as each addition and multiplication operation is performed on it. When this noise grows beyond a critical threshold level, the result of the homomorphic computation is destroyed and cannot be recovered after decryption.

In this same work, Gentry also showed how a technique called bootstrapping could overcome the noise accumulation problem. Bootstrapping helps reset the noise to a lower level so that further computations can be performed. Using this technique with his initial SHE scheme, Gentry described the first Fully Homomorphic Encryption scheme. However, bootstrapping has high latency, and this high latency remains one of the key bottlenecks in the adoption of HE.


1976
Diffie-Hellman key exchange invented
1977
RSA public key encryption invented
1984
First quantum key distribution scheme, BB84
1991
Public key encryption program PGP released
1994
SSL protocol released
2009
First Fully Homomorphic Encryption scheme
DES becomes FIPS standard
1978
Homomorphic encryption proposed
1985
Elliptic-curve cryptography invented
1995
SHA1 hash algorithm published
2000
AES replaces DES
2001
SHA2 published

A timeline of encryption milestones leading to the development of FHE

HE VS. OTHER TECHNIQUES

As the importance of data privacy has moved to the forefront in recent years, various privacy-preserving techniques have gained prominence: multiparty computation, zero-knowledge proofs, differential privacy, and trusted execution environments. The use cases for each of these differ substantially and contrast with HE.

Unlike multiparty computation and zero-knowledge proofs (both cryptographic solutions), HE requires no ongoing interactions between the parties providing data or computation. In HE, a client encrypts the data, sends it to the cloud for processing, and, after that point, can remain offline while the cloud server carries out computations. No key exchange between the client and the cloud is necessary. Once the computations are finished, the cloud can send the encrypted data back to the client. In contrast, both multiparty computation and zero-knowledge proof techniques require the client

ⓘ

[Mercè Crosas and James Honaker discuss their approach to multiparty computing and differential privacy in the article "Voyage into the open Dataverse," RHRQ 2:2, August 2020.]

Unlike multiparty computation and zero-knowledge proofs, HE requires no ongoing interactions between the parties providing data or computation.

to actively participate in the process of computation and remain online for the entire duration of the computation.

With trusted execution environments like Intel SGX/TDX and AMD SEV/SNP, the data needs to be decrypted within the secure enclave before it is used for computation, requiring the data owner to share the decryption key with the server in most cases. If the cloud is not completely secure, maintaining the security of the decryption key itself could be challenging. Additionally, any attacker that can exploit the specific trusted execution environment to read the isolated memory area would also be able to read the data. However, with HE, the data owner does not need to share the secret keys with the cloud server, and any read of the memory still reveals only encrypted data.

In contrast, differential privacy is a non-cryptographic solution to data privacy. It adds random noise to the private dataset so as to conceal the actual values. This can preserve the privacy of individual data points in a mathematically rigorous way when working with aggregated data, such as US Census data. While FHE enables computation outsourcing to a third-party cloud, differential privacy is typically adopted for federated machine learning operations.

HOW DOES HE WORK?

In 2009, when FHE became feasible, the option to compute encrypted data with FHE was available only for Boolean operations, which describe logic gates, the building blocks of circuits. All computations were first expressed as Booleans before they could be evaluated homomorphically. Translating mathematical operations into Booleans was known as mapping to a homomorphic circuit. However, with the latest FHE schemes, conversion to a homomorphic circuit involves translating the plaintext compute model to a combination of primitive FHE operations such as addition and multiplication. In addition, depending on the specific FHE scheme, the data involved must first be encoded as polynomials.

There is a hierarchy that describes HE schemes and their capabilities. The least capable scheme is Partially Homomorphic Encryption, which supports the evaluation of either addition or multiplication, but not both. (While division and subtraction are possible indirectly in HE, they are more complex, and most descriptions of HE schemes speak in terms of addition and multiplication operations.) SHE and Leveled FHE schemes are more capable intermediate categories and can compute many additions and a small, predetermined number of multiplications on ciphertexts.

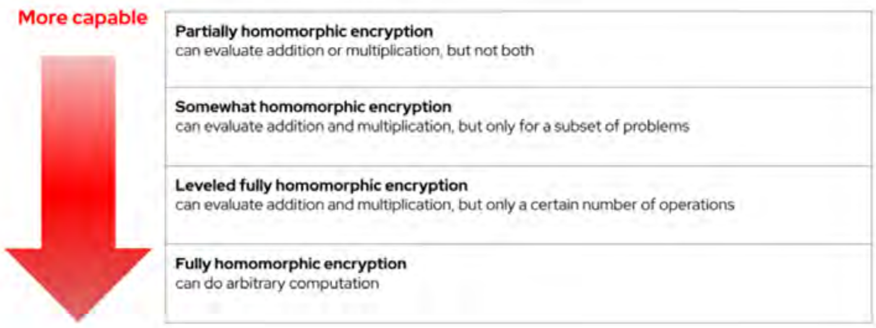
FHE schemes, considered the “holy grail” of HE, allow the evaluation of multiple types of operations with no maximum on multiplication operations. In other words, they enable an infinite number of calculations on ciphertext while still producing a valid result that can be decrypted. For most existing HE schemes, the cap on multiplication operations is the main practical limitation in performing computations over encrypted data. Recall that the cap can be lifted by bootstrapping, but this introduces a very high level of latency that is unacceptable in most real-world scenarios.

HARDWARE ACCELERATORS

Only by narrowing the performance gap will wide-scale adoption of FHE be feasible. Existing research in the field of FHE tries to reduce the execution time of operating on encrypted data by either optimizing algorithms or the implementations themselves. To this end, several FHE acceleration efforts are based on CPUs, GPUs, and ASICs. The CPUs and GPUs have shown they cannot bridge the performance gap, and ASIC solutions are expensive, with large on-chip memories.

This is why Prof. Joshi's team at BU proposed acceleration based on field programmable gate arrays (FPGAs) as one of the viable solutions that could provide practical performance for FHE while being affordable. Surprisingly, there was limited work on the acceleration of FHE schemes on FPGAs prior to their efforts, despite the obvious gap in the context of the problem space. FHE workloads are inherently parallel, but existing commodity platforms like CPUs are not good at exploiting this parallelism. GPUs are up to the challenge of exploiting the existing parallelism in FHE workloads, but they have massive amounts of floating-point compute units that remain idle since FHE operations are integer-only computations. Moreover, neither CPUs nor GPUs provide native support for modular arithmetic operations, which is the main compute bottleneck in FHE. Lastly, both of these platforms fail to meet the high memory bandwidth requirement of the FHE workloads.

On the other hand, while ASIC solutions will always have higher performance than FPGA solutions, ASIC solutions are a lot more expensive.



Partially homomorphic encryption
can evaluate addition or multiplication, but not both

Somewhat homomorphic encryption
can evaluate addition and multiplication, but only for a subset of problems

Leveled fully homomorphic encryption
can evaluate addition and multiplication, but only a certain number of operations

Fully homomorphic encryption
can do arbitrary computation

In addition, ASIC solutions are not futureproof and will require a non-trivial amount of redesign as the FHE algorithms evolve in the future.

The performance, cost, and flexibility of FPGAs suggested to Prof. Joshi's team that FPGAs would provide a sweet spot between CPUs/GPUs on one end and ASICs on the other for their FHE acceleration work. They determined that an FPGA solution could significantly outperform both CPU and GPU implementations of FHE. Moreover, FPGAs are highly accessible even today to the general public and can be deployed immediately for under a dollar per hour. By enabling competitive levels of performance with the same availability as a high-end GPU, FPGAs are the most viable option for near-term hardware FHE acceleration in the cloud.

Xilinx's Alveo boards and Intel's stratix FPGAs are already available in some cloud environments today.

IMPRESSIVE SPEEDUP

In their research, the team's first step was to choose an appropriate FHE scheme based on their underlying application of logistic regression (LR) model training, specifically an image-classification-based healthcare application. They chose the CKKS scheme (an acronym of the creators' names) because it supports operations on floating-point data (this data is translated into polynomials requiring integer-only computations once encrypted). Using this scheme, they designed and implemented various primitive homomorphic operations for their specific application, including addition, multiplication, rotation, conjugation, key switching, and bootstrapping. All these operations were customized for their U280 FPGA implementation so that they incur low hardware-resource utilization and low latency.

The next task was mapping the actual LR application using these building blocks. For FHE-based computing, there are two critical steps to follow. First, they had to figure out a way to efficiently transform the dataset into ciphertexts according to the rules of the CKKS scheme, and second, they had to map the actual equation into what is known as the homomorphic circuit. They had to carefully consider how much on-chip data their FPGA could handle and concluded that they could pack their entire training dataset within 84 ciphertexts.

Once they were able to map the LR equations to the homomorphic circuit, they saw that it took on average 0.1 seconds to perform an iteration of training with the LR model, which is about 458 times faster than existing CPU implementations and about nine times faster than the existing GPU implementations of this same set of homomorphic operations. The FHE accelerator takes 3.09 seconds to train a logistic regression model with 30 iterations. The same training takes about 0.12 seconds on plaintext data using a CPU. Effectively, the slowdown is 26x—much less than other current HE libraries not using an accelerator, where the slowdown can be around 1,000,000x.

FUTURE POSSIBILITIES IN THE CLOUD

For the use of an FHE FPGA accelerator in the cloud, the host CPU on the cloud could offload the FHE computation, in the form of a binary file, and the encrypted data to the accelerator. The accelerator could then perform homomorphic computations by running the binary and, once the computations are complete, send the results back to the host CPU, which would return them to the client. This would be possible on any cloud hosting the appropriate FPGA shell.

From the data owner's point of view, taking advantage of FHE in the cloud with an FPGA accelerator would be a two-step process. First, an expert would need to create the mapping from the plaintext application to the correct homomorphic circuits. Each application should use its own customized circuits to retain the performance benefits of the hardware accelerator. Second, once the circuits are ready, the data owner would encode the data into polynomials, encrypt it using existing libraries such as SEAL or PALISADE that support the CKKS FHE scheme, and upload it to the cloud. The length of time the computation would need in the cloud would depend on the application and dataset, and would need to account for the slowdown of the homomorphic computation. Once the data owner receives the encrypted results from the cloud, they would be able to perform decryption and decoding to recover the results in plaintext format.

USE CASES AND ONGOING RESEARCH

FHE is a closer fit for some types of problems than others due to HE's dependence on mapping to polynomials and circuits in a specific manner. For example, FHE is a good fit for logistic regression machine learning models, whereas it is less of a good fit for machine learning models used for natural language processing.

Using Prof. Joshi's team's work, the ability of FHE to perform processing on encrypted data with a less costly slowdown has the potential to solve major business challenges faced by companies across all industries. Some of the examples include:

- Private data analytics for marketing, targeted advertising, rare disease diagnosis, genome analysis, drug research, and many more
- Regulatory compliance, such as GDPR (EU General Data Protection Regulation) and HIPAA (US Health Insurance Portability and Accountability Act)
- Private information retrieval through secure database queries
- Secure supply chain management
- Secure bioinformation authentication

The team's work is part of the project "Privacy-preserving cloud computing using homomorphic encryption," which was a recipient of the 2021 Red Hat Collaboratory Research Incubation Award. The team successfully deployed the hardware accelerator on an FPGA node in the Open Cloud Testbed in June 2022, and they plan to demonstrate its use to perform neural-network-based classification of encrypted image data by the end of this year.

About the Authors

Rashmi Agrawal is a PhD candidate in Electrical and Computer Engineering at Boston University. She is a member of the Integrated Circuits and System Group and is advised by Prof. Ajay Joshi. Her research interests include designing efficient architectures and co-processors for post-quantum cryptography and hardware acceleration of privacy-preserving computing using fully homomorphic encryption.

Lily Sturmann is a senior software engineer at Red Hat in the Office of the CTO in Emerging Technologies. She has primarily worked on security projects related to remote attestation, confidential computing, and securing the software supply chain.

A summer in Europe: US students thrive in open source research opportunities abroad

What do visualizing microservice architecture and the Punkva Caves have to do with each other? Two summer visitors to Brno can explain.

by Mia Gortney and Patrick Harris

About the Authors

Mia Gortney is studying Computer Science at Baylor University. She received the Computer Science Scholarship Award in Spring 2022 and has been featured on the Baylor University Dean's List. Her research interests include static and dynamic code analysis

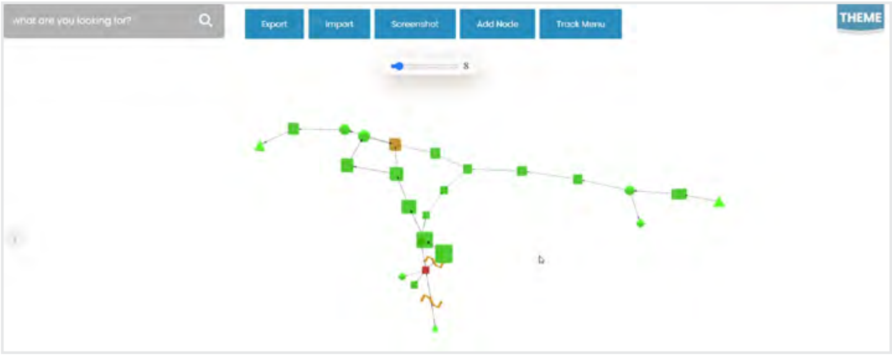
Patrick Harris is studying Computer Science at Baylor University. He has been recognized on the Baylor University Dean's List and received scholarship awards including the Baylor Computer Science Scholarship and the Baylor Association of Computing Machinery Scholarship. His research interests include the visualization of distributed systems and the security of computer systems.

Summer 2022 marked the fourth time professor Tomáš Černý brought promising undergraduate students from his home institution, Baylor University in Waco, Texas, USA, to the Czech Republic for an innovative research program sponsored in part by the US National Science Foundation. Students split their time between Czech Technical University in Prague and the offices of Red Hat Research in Brno.

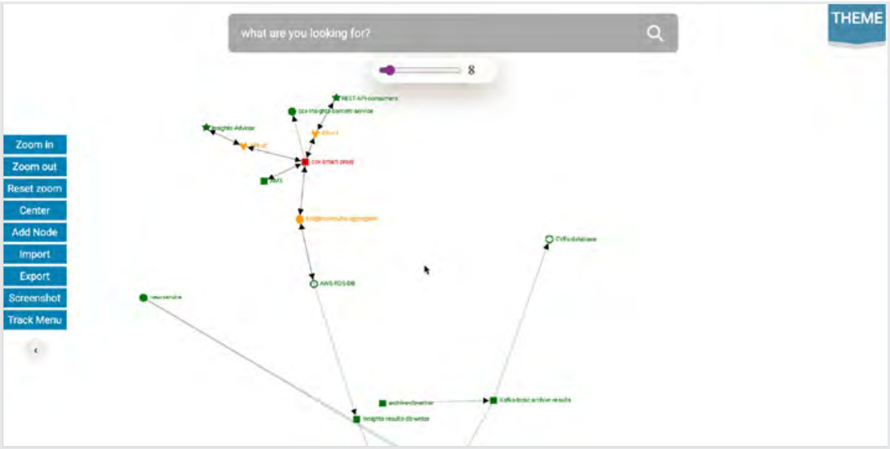
Mia Gortney and Patrick Harris are student researchers associated with the Baylor University Department of Computer Science. Mia and Patrick are both Texas natives and had never left the United States before this program. They were offered this opportunity by Dr. Černý and did not hesitate to accept. In this article, they describe their experience with Red Hat, the Czech Republic, and open source research.

Our first month in Czechia was spent in Brno, working alongside Red Hat employees. The main focus of our research was visualizing microservice architecture from the dynamic perspective. While in Brno, we focused on composing research into a mapping study to illustrate work that has been done on the topic. We worked with Red Hat quality assurance engineer Pavel Tišnovský and Dr. Černý to compile a list of articles, which were then filtered based on inclusion and exclusion criteria. After reading through these articles, we more clearly defined our topic into three subtopics: dynamic analysis techniques, tools that accomplish these techniques, and visual perspectives generated by these tools. Using the knowledge acquired through our research, we composed a literature review that has now been submitted to ICSOC and IEEE Access. We are currently waiting to hear from IEEE Access with hopes of getting published.

Throughout this research process, Mr. Tišnovský and Red Hat as a whole were extremely helpful to us. They provided access to their facility and research knowledge from an industry perspective. Through them, we were invited



3D representation of a microservice-based System using Three JS



2D representation of a microservice-based system using D3 JS

[Dr. Černý spoke to RHRQ about microservices research and his student research initiative in an interview in RHRQ 4:2, From Brno to Waco: On cross-cultural exchange, microservice evolution, and quality assurance.]

to attend DevConf.CZ 2022, a conference centered around software development. We also got the opportunity to listen to a research presentation given by Dr. Černý to the Red Hat staff. Being in this work environment for the first month was extremely beneficial to us as student researchers. We gained more insight into microservice systems and applied that knowledge to a mapping study and a prototype.

Toward the end of our first month and throughout our second month in Prague at Czech Technical University, we developed a visualization prototype. The main objective of the prototype was to develop a framework to visualize microservice architecture in a holistic and interactive manner.We wanted to frame the visualization in a lightweight web application, so JavaScript seemed like a good choice. Utilizing JavaScript allowed us to rely on visualization libraries that served as the foundation for our prototype. We chose D3 as it provided a two-dimensional force-directed graph and Three JS because it provided the capabilities for a force-directed graph in three-dimensional space.

Starting with just a loose connection of nodes and links supplied by the libraries, we added and built many custom features to better handle the visualization of microservice-based systems. Within a few weeks, we had implemented interactive searching, different node shapes to differentiate between system parts, dynamic coloring to highlight areas of high coupling, information boxes to describe each node, a right-click context menu to better interact with nodes, and even features aimed at predicting the impact of changes throughout a system's evolution.

One of the most important steps was collaborating with our other peers to create a high-level JSON schema that can describe a service-level dependency view of a microservice system. This schema serves as the input for our prototype, translating a static description of a system into a dynamic, interactive graph. Ultimately, by using the knowledge gained from the mapping study, we were able to quickly develop a proof-of-concept prototype that abstractly visualizes a microservice-based system and addresses gaps we identified in current research.

Since returning to the United States, we have continued our research into visualizing microservice systems from the dynamic perspective. We are currently refining an institutional review board (IRB) study for approval, centering around whether a two-dimensional or three-dimensional view is more beneficial for the user when visualizing a microservice system. Once the study is approved, we are going to conduct it with 30 experts in microservice systems to discover which view we

2D representation of Baylor students visiting popular tourist sites in Western and Central Europe

should focus on and refine further. We are also redeveloping our three-dimensional prototype so the visual is cleaner and the features are more extended. We are working with other IRES (International Research Experience for Students) research groups that focus on architectural languages, provenance tracking, and anti-pattern detection to incorporate their tools into our own prototype. By the end of the year, we should have a new and improved prototype to effectively visualize distributed systems. Anyone interested can contact us at Mia_Gortney1@baylor.edu and Patrick_Harris3@baylor.edu.

Over the course of the summer, we had the opportunity to visit many incredible places. During the week, we would work our six-hour day and then explore Brno or Prague in the evenings. While in the Czech Republic, we visited the Punkva Caves and participated in a paintball game at Army Park Orechov outside Brno. We also visited the Astronomical Clock and several other historical buildings in Brno and Prague. On the weekends, we would travel by plane or train to other countries. In our first weekend, we visited Budapest, Hungary, and Vienna, Austria, back to back. The following weekend, we went to Bratislava, Slovakia, and later we traveled to Kraków, Poland, and Munich, Germany. Towards the end of the summer, we visited London, Zurich, and Paris. We also visited three places in Italy: Milan, Venice, and Rome. Without this research opportunity presented by Baylor University in collaboration with Red Hat, there is no way we would have been able to visit all of these countries in such a short amount of time. The amount of culture we got to experience all over Europe is something we will never forget.

The research opportunity this summer provided us with valuable practical experience in our field and new cultural experiences outside the United States. Neither of us had previous experience with microservices, but delving into research in this area allowed us not only to learn about a relevant field in the industry right now but also to solve real-world issues and ultimately help advance microservice technologies. Developing a prototype allowed us to enrich our JavaScript skills, which has proven very useful as we are using JavaScript for the front end of our group project in Software Engineering II this semester. Finally, it is important to highlight how key the cultural immersion was in expanding our worldviews and illustrating how unique and beautiful different areas of the globe are. We had an absolute blast this summer, and we are very thankful to Dr. Černý, Mr. Tišnovský, Red Hat, Czech Technical University, and all the amazing people we met during our travels for the experience of a lifetime.

Column

Europe RIG increases collaboration with the European Commission to drive innovation for addressing global challenges

About the Author
Matej Hrušovský has been with Red Hat for more than nine years, seven 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.

by Matej Hrušovský

The year 2022 was exceptional for Red Hat Research in a multitude of ways. After more than two years of only virtual gatherings, we successfully organized Red Hat Research Day Europe, an international in-person event held in Brno, Czech Republic. We further consolidated and aligned our team's goals with the goals of the company and expanded our partnerships in multiple directions. But one of the most stellar successes happened in the sphere of EU-funded research projects.

Horizon Europe is an EU-funded program that facilitates collaboration across companies, academic institutions, and research centers. Its goal is to strengthen the impact of research and innovation in developing, supporting, and implementing EU policies while tackling global challenges. The program also supports creating and more impactfully dispersing excellent knowledge and technologies.

These aren't the first EU-funded projects that Red Hat has taken part in. Most of the early projects, however, were managed locally, usually by the engineering teams themselves. This is also what makes 2022 noteworthy: all the projects chosen by the EU Commission for funding have been led by a team of Red Hat Research members and various engineers from all over EMEA.

We hardly expected that almost all the projects we applied for would receive an EU grant—but they did. This was a big challenge for the entire team because it was tough to predict the final result. We managed to scale our duties and responsibilities in cooperation with the massive amount of partners involved to make this dream a reality. We worked with more than 100 partner organizations worldwide, including the likes of Intel, IBM, Atos research, Huawei, Caixabank, Siemens, and Telefonica, among others.

We have a strong commitment with the European Commission to make sure these projects end as they are supposed to. All of us involved in the projects are accountable for fostering a culture of innovation, creativity, and engagement. This is a culmination of many ongoing Red Hat Research efforts, and it will now become a long-term relationship between Red Hat, academic institutions, industrial partners, and the European Commission.

Here is a brief description of the projects that will start in 2022-2023:

INCODE: Programming platform for intelligent collaborative deployments over heterogeneous edge-IoT environments

The INCODE programming platform intends to provide the following:

- An orchestration umbrella. By tightly integrating state-of-the-art IoT, edge/cloud computing, and networking platforms, the umbrella would manage end-to-end environment resources according to end-user requests and application content.
- A programming environment for applications. The environment would speed up development and deployment processes on large swarms of devices, split the application workload tasks efficiently into microservices, deploy them over heterogeneous edge-IoT infrastructures, and provide fast reconfiguration of required resources according to the application needs and IoT device status.
- A secure and trusted framework for edge-IoT deployments. The framework would provide IoT device and system authentication, data management, and application deployment; device attestation and resource availability monitoring; and guarantees of integrity with respect to the deployed applications, the shared data, and the generated data.

ICOS: Towards a functional continuum operating system

The main objective of project ICOS (IoT2Cloud Operating System) is to design, develop, and validate a meta-operating system for a continuum by addressing the challenges of:

- Device volatility and heterogeneity, continuum infrastructure virtualization, and diverse network connectivity
- Optimized and scalable service execution and performance, as well as resource consumption, including power consumption
- Guaranteed trust, security, and privacy
- Reduction of integration costs and effective mitigation of cloud provider lock-in effects in a data-driven system built upon the principles of openness, adaptability, data sharing, and a future edge market scenario for services and data

Green.Dat.AI: Energy-efficient AI-ready data spaces

The long-term vision of focusing on efficiency is to allow computing to move from data centers to edge devices, making AI accessible to more people, for example, by shifting computation from the cloud to personal devices to reduce the flow and potential leakage of sensitive data or by enabling processing data on the edge to eliminate transmission costs, lead to faster inference with a shorter reaction time, and drive innovation in applications where these parameters are critical.

CONNECT: Continuous and efficient cooperative trust management for resilient CCAM

The vision of CONNECT is to address the convergence of security and safety in cooperative, connected, and automated mobility (CCAM) technology. CONNECT will assess dynamic trust relationships and define a trust model and trust reasoning framework based on which involved entities can establish trust for cooperatively executing safety-critical functions. This will enable cybersecure data sharing between data sources in the CCAM ecosystem with no or insufficient preexisting trust relationships and allow trustworthy outsourcing of tasks to multi-access edge computing (MEC) and cloud technologies. Beyond the needs of functional safety, trustworthiness management should be included in CCAM's security functionality solution for verifying the trustworthiness of transmitting stations and infrastructure.

CODECO: Cognitive decentralized edge cloud orchestration

CODECO proposes the following assets:

- Open cognitive toolkits and smart apps integrating elastic and advanced concepts to manage, in a smart and flexible way, containerized applications across edge and cloud
- A developer-oriented open source software repository to be available in an early stage of the project, thus allowing for early exploitation of initial, advanced results and better adaptation throughout the project's lifetime
- Training tools to support the development of services based on the CODECO framework
- Use cases across four domains (smart cities, energy, manufacturing, and smart buildings) as the basis for experimentation and demonstrations
- Open calls and multiple community events based on the various use cases and including varied CODECO stakeholders

CODECO integration into the large-scale EdgeNet experimental infrastructure to assist in the building of experimentation and novel concepts by the research community

AC3: Agile and cognitive cloud edge continuum management

The key objective of the AC3 project is to devise a novel Cloud Edge Computing Continuum (CECC) manager that heavily relies on AI/ML to manage the lifecycle of applications and the underpinning IT and networking resources while ensuring that it operates as a cognitive system on top of a federated infrastructure composed of cloud, edge, far edge, and data sources from different stakeholders. The envisioned CECC architecture would ease resource federation by relying on a well-defined, secured, trusted, and open API to guarantee interoperability and the seamless deployment and management of applications over the federated infrastructure.

AERO: Accelerated European cloud

AERO has the single mission of enabling the future heterogeneous EU cloud infrastructure. Towards this, it will develop—to a high technology-readiness level —all components necessary to achieve out-of-the-box heterogeneous execution of the cloud ecosystem on the EU processor. The outcome will be a set of compilers, runtime systems, operating systems, system software, and applications that can leverage the underlying capabilities of the future EU cloud equipped with GPUs, FPGAs, and other accelerators.

CHESS: Cybersecurity excellence hub in Estonia and South Moravia

The Cybersecurity excellence hub in Estonia and South Moravia (CHESS) will bring together leading research and innovation institutions in both regions to build connected innovation ecosystems and address one of the most critical issues confronting Europe today: cybersecurity. The strategizing, skills building, and pilot research and innovation will cover the totality of the cybersecurity field, with special attention to six challenge areas: internet of secure things, security certification, verification of trustworthy software, blockchain, post-quantum cryptography, and human-centric aspects of cybersecurity.

ⓘ

[Find out more about participation in EU-funded research schemes in the story "Look to the Horizon: Europe's increased focus on funding open source research is creating new opportunities," RHRQ 4:1, 2022.]

Research project updates

Each quarter, *Red Hat Research Quarterly* highlights new and ongoing research collaborations from around the world.

This quarter we highlight a few collaborative projects from the United States at Boston University, Emory University, Tufts University, Northeastern University, and Columbia University, and in Germany at TU München. Contact academic@redhat.com for more information on any project described here or visit the Red Hat Research Project Directory *on research.redhat.com.*

PROJECT: Characterizing microservice architectures

ACADEMIC INVESTIGATORS:
Prof. Raja Sambasivan (lead) and PhD students Darby Huye and Lan Liu (Tufts); Prof. Avani Wildani and PhD student Vishwanath Seshagiri (Emory)

RED HAT INVESTIGATORS:
Not specified (industry participants are anonymous due to the nature of this research)

Prof. Raja Sambasivan and his students collaborated with Prof. Avani Wildani of Emory University and her students to complete a user study with microservice developers. The study aims to identify the variety of industrial microservice architectures in use by developers in industry and compare those architectures to those available in current open source testbeds such as DeathStarBench, TrainTicket, and BookInfo. Investigators used codebase analyses, literature reviews, and developer interviews to probe industrial microservice designs in detail.

The researchers found that industrial architectures vary greatly from testbeds and that this may invalidate research assumptions for work using existing testbeds. The teams hope to broaden open source testbeds and to continue identifying and classifying microservice architectures in future work. This work will also benefit their previous work on tools and data collection for large-scale traces in realistic distributed system computing topologies and their work on open telemetry with Red Hat and OpenInfra Labs. Their article "SoK: Identifying mismatches between microservice testbeds and industrial perceptions of microservices" was published in the *Journal of Systems Research* 2:1, June 2022.

PROJECT: LEGO for 6G: A modular open access end-to-end network architecture to enable B5G/6G research

ACADEMIC INVESTIGATORS:
Abhimanyu Gosain, Senior Director, Institute of Wireless Internet of Things; Prof. Tommaso Melodia, IWIOT Institute Director; Michele Polese, Principal Research Scientist (Northeastern)

RED HAT INVESTIGATOR:
Dan Winship

The IoT team's work evolved from 5G to 6G since the last research update, suggesting a roadmap for leveraging public-private partnership research testbeds to influence the 6G vision and accelerate the entire lifecycle of research and development, manufacturing, standardization, and market readiness for 6G. Following a recent press release announcing the formation of the Open6G Cooperative Research and Development Center at Northeastern, the group is exploring topics such as spectrum access and exploitation, Open Radio Access Network (O-RAN) architectures, AI/ML for inference and control, and mmWave and Terahertz systems.

Traditional vendors treat the Radio Unit (RU) and Distribution Unit (DU) of a Radio Access Network (RAN) together as a black box. The international testbed where Gosain's group and collaborators work, which is now running OpenShift for experiment and node deployments, allows separate experimentation with RU and DU components. This unlocks a lot of interesting possibilities for optimizing and controlling RANs. Abhimanu Gosain presented this work at the September Research Day event in Brno; slides from the presentation and a video recording of the session are available at the Red Hat Research website.

PROJECT: Near-data data transformation

ACADEMIC INVESTIGATORS:
Prof. Manos Athanassoulis (PI), Prof. Renato Mancuso (Co-PI), and PhD students Shahin Roozkhosh and Tarikul Islam Papon (BU); Denis Hoorneart, PhD Student (TU München)

RED HAT INVESTIGATORS:
Uli Drepper and Ahmed Sanaullah

This team re-examined their previously implemented software-hardware co-design approach, which pushed data transformation closer to memory from a real-time systems perspective. By doing realistic prototyping with CPU + FPGA platforms, the team has been able to allow efficient and cache-friendly access to large data objects by moving only relevant data items from target memory. This compressed the working-set size, and therefore the cache footprint, and reorganized complex memory access patterns into sequential access with predictable patterns.

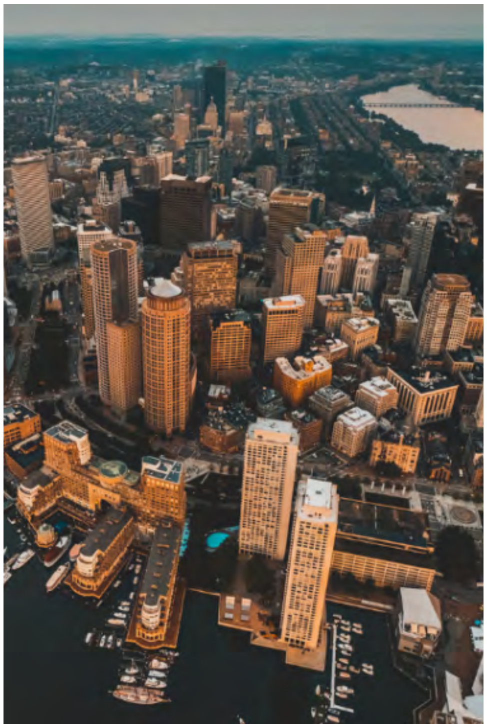
The team dubbed this approach Programmable Logic In-the-Middle (PLIM), presenting it as an important part of an Open Hardware Initiative Series at DevConf.US 2022, in both a talk and a Q&A panel discussion. Prof. Mancuso also presented related research ideas at the October virtual Red Hat Research Days event "Can we control time? Toward knowledge-driven system management to control timeliness." A video recording can be found in the Past Events section of the Red Hat Research website.

PROJECT: Automated detection of memory safety vulnerabilities in Rust

ACADEMIC INVESTIGATORS:
Prof. Baishaki Ray and PhD student Vikram Nitin (Columbia)

RED HAT INVESTIGATORS:
The (Anne) Mulhern and Sanjay Arora

When Rust code is compiled, it goes through a High-level Intermediate Representation (HIR) and a Mid-level Intermediate Representation (MIR). This project team has implemented a hybrid analysis that combines information from the HIR and the MIR to detect code patterns that may cause memory safety violations, such as incompatible lifetime dataflow conditions. The team has been using the system, implemented as a subroutine within the RUDRA project, to examine Rust code in open source repositories. The team is currently analyzing results from this process and expects to make their code available in a publication soon. For more on the team's work, see their virtual Research Day event in the Past Events section of the Red Hat Research website. For more information about RUDRA, see Yechan Bae et al., "RUDRA: finding memory safety bugs in Rust at the ecosystem scale," in Proceedings of the *ACM SIGOPS 28th Symposium on Operating Systems Principles*. (2021).



intel AI
AI ON INTEL®
intel XEON PLATINUM inside™
NOW BUILD THE AI YOU WANT
ON THE CPU YOU KNOW.
Learn more at ai.intel.com