RHRQ

Bringing great research ideas into open source communities

VOYAGE into the open dataverse

Open source cloud operations

Sharing hardware safely with Elastic Secure Infrastructure

+

Don't blame the developers

James Honaker and Mercè Crosas on the privacy balancing act

Red Hat
Research Quarterly
Volume 2:2 | August 2020 | ISSN 2691-5278

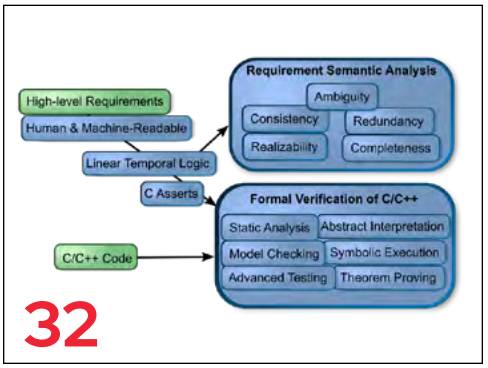

DELL Technologies
LET'S MAKE
SUSTAINABLE
REAL
Already a leader and pioneer in electronic waste recycling, Dell Technologies commits to regenerating our environment, reducing our impact and relying on renewable resources. Now, and in the generations to come.
DellTechnologies.com

RESEARCH QUARTERLY

VOLUME 2:2

Red Hat

Table of Contents



Departments

Features

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

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

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.

From the Director

Reproducible research

If a tree falls in the forest, but you can't reproduce it, how do you know if it made a sound or not?

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.

Readers of this magazine should, I would think, be familiar with the basic process that constitutes the scientific method: decide on a hypothesis based on one's experience and ideas of what "ought" to be true; design an experiment that may supply data to support that hypothesis; analyze the data in the hopes that it is conclusive. The "conclusive" nature of the data may either support or disprove the hypothesis, and either result is good in that both the support and the negation of a hypothesis are a new piece of knowledge.

There is, of course, a follow-on action to this process that is of critical importance: share the experiment, the data, and the analysis, such that others can reproduce the result and verify it. The requirement that you share everything for reproducibility, although not a formal part of the method, is perhaps the most critical step. Without reproducibility, science cannot advance beyond a single lab, and may in fact not advance at all. The community of researchers—the grandparent of the open source development communities many of us participate in today—is where the true value of science is realized, and if there is such a thing as progress in the world this community is surely at the heart of it.

Unfortunately, for scientific research that involves people—as is often the case in medicine, the social sciences, or artificial intelligence—sharing experimental data may be impossible due to privacy or similar concerns. This not only slows scientific progress but can lead to false results due to innocent mistakes or worse. Our interview in this issue features two people—Dr. Mercè Crosas and Dr. James Honaker—whose work is devoted to making experimental datasets consistently and universally available. Their work enables researchers developing statistical techniques to glean knowledge from data without seeing the raw data itself. We think these techniques are critical not only for the advancement of science but for open source development in AI, where training data for deep learning is a key tool. I think you'll find the interview fascinating.

Speaking of innocent mistakes, have you ever clicked "OK" on a threatening-looking security warning without really reading it? I know I have, and felt vaguely nervous about it every time. Don't feel bad, though: Martin Ukrop's work on "usable security" shows that seemingly minor improvements to the text of common

security warnings makes an outsize difference in whether people respond to them appropriately. Clicking through an obscure warning that doesn't make any sense may not be entirely your fault after all.

> We think these techniques are critical not only for the advancement of science but for open source development in AI...

While we're discussing open data, I want to highlight our call to action in this issue to participate in an experiment we are launching with Boston University on Open Operations. We intend to operate a cloud in partnership with BU and Harvard University that will allow the collection and analysis of all the operational data about that cloud (with the express permission of the users, of course). It is time we made operations at scale into an open discipline, like open source software. See the article on Operate First to learn more.

Finally, on a personal note, all of us at Red Hat Research feel very fortunate to be largely unaffected by the pandemic we are dealing with in our various countries. We hope you readers have been similarly fortunate. If anything positive comes out of this, perhaps it will be some level of return to the idea that hypothesis, experiment, and proof or disproof are the only way to be mostly sure of anything.

Red Hat Research Days coming this fall

by Gagan Shantha Kumar and Heidi Dempsey

During Red Hat Research Days, researchers, Red Hatters, technologists, and students come together to discuss exciting new research developments. These developments will change the way we build and use computers, clouds, and the crucial data we entrust to those systems. Animated conversations in and around Research Days often turn into new projects with industry and academic researchers.

This year, as we all collaborate remotely, Research Days will also go virtual. We're planning to expand Research Days in the United States to include a series of conversations between researchers and Red Hat experts that will give remote participants a deeper look into the work that's highlighted for the event. Our hope is that these conversations will capture the excitement and challenge of exploring new research in depth, while giving participants a chance to share questions and new ideas for collaborations on other days. Schedules for these fall conversations and more details on the September 22, 2020, agenda will be shared on the Red Hat Research website (research.redhat.com). For a preview of one of the Research Days topics, see the interview in this issue with Mercè Crosas and James Honaker of Harvard discussing differential privacy and open, reproducible data repositories.

Research Days discussions and presentations will include experts from many different universities and research groups around the United States. (Research Day Europe, https://research.redhat.com/blog/events/research-day-brno-2020/, held in January, largely covered European work.) This year, we're exploring research to improve privacy and security, make experimentation and system execution more reproducible, and enhance the performance of cloud systems. Researchers from Boston University, Harvard, Columbia, Yale, Carnegie Mellon, North Carolina State University, University of California–Santa Cruz, University of Illinois–Champaign-Urbana, and the University of Chicago will be participating, along with several Red Hat experts.

To learn more about Research Days and how you can be part of this experience, please visit research.redhat.com/research-day.

Why you should (virtually) attend Devconf.US

We know you're being deluged with event invites and it's hard to decide where you should spend your time. Devconf.US has a unique experience to offer. Here's why you should register for Devconf.US—for free!

by Gordon Haff

We've refined Devconf over more than a decade. The annual event began in 2009 in Brno, Czech Republic, where Red Hat has a large engineering office. A few years ago, we added an event in the Boston area, where Red Hat has another large engineering presence. This historically close connection to Red Hat engineering locations, plus local universities, means that attendees have great access to those working in upstream open source communities.

How does that access work now that the event is virtual? Like many organizations, we've probably learned more about running virtual events this year than we did in the prior decade. For example, this year's Red Hat Summit had over 80,000 registrations, a huge increase over prior years' physical events. We're applying what we've learned to Devconf.US, including event chat, live Q&A for each session, Ask the Experts, and even the event party. It's hard to fully replicate the "hallway track" online, but you'll be pleasantly surprised by the quality of 1:1 and small group interactions that we're lining up for this virtual event.

No planes, trains, or automobiles required. The upside of a virtual event is that it's easy to drop in and give it a try. If the time commitment or the travel budget needed to attend an in-person Devconf has scared you off in the past, this is a great opportunity to sample the content and interact with the speakers. We hope doing so will have you looking forward to coming back when we can return to running physical events.

> Devconf has always been about open source and community projects.

No product pitches. Devconf has always been about open source and community projects. Not big expo halls. Not product content. This overarching philosophy extends to the virtual Devconf.US this year. You'll also get a chance to listen to—and ask questions after—relevant talks where students and professors will discuss their active, ongoing research in areas like emerging tech.

Devconf encourages and supports new speakers and attendees. It's always been important to the organizers for Devconf to be especially welcoming, not just to junior speakers, student speakers, and underrepresented speakers, but really to anyone, speaker *or* attendee, who is not necessarily a regular on the conference circuit. To this end, we have mentors and other support for anyone who might be uncertain about how to best participate and enjoy the conference experience. Devconf is the first conference for many speakers and attendees, but many go on to present and participate in the largest industry events.

Are you interested in open source? Then Devconf.US has something for you. Topics cover everything from cloud native app dev to security to AI/ML to software quality to hardware innovations and more. We hope you take advantage of this virtual event to discover open source activities that excite you. Register at https://www.devconf.info/us. And if you mention us on social media, please tag #DefineFuture.

Feature

How expensive is it to crack a password derived with Argon2? Very

Passwords made are to be memorable, so they are not usually secure enough for encryption software. That's where derivation functions come in, transforming a password into a more suitable cryptographic key. Memory-hard functions—functions that cost a significant amount of memory to evaluate—are especially useful to mitigate time-memory trade-off attacks. Here, we describe research on Argon2, one such memory-hard function.

by Vojtěch Polášek

About the Author

Vojtěch Polášek, in his own words: "I am a blind Linux guy interested in information security. In my free time I explore new technologies, I play blind football, and I am involved in various projects connecting visually impaired and sighted people together."

This article summarizes my research while a master's student at the Masaryk University Faculty of Informatics. I simulated an attack on a disk encrypted with the LUKS2 encryption scheme using Argon2, the 2015 winner of the Password Hashing Competition (password-hashing.net), as the password-based key derivation function (PBKDF). During this simulated attack, I collected Argon2 parameters benchmarked by Cryptsetup software. The attack simulation ran over both CPUs and GPUs and allowed me to estimate the costs to an attacker using either physical hardware or on-demand allocation of computing resources in the cloud.

The results were astonishing: it can take thousands of machines and hundreds of millions of dollars over ten years to crack an eight-character LUKS2 password using Argon2.

PASSWORDS NEED TO BE RESILIENT

Organizations are putting more pressure on all of us to create more secure passwords, but this is very difficult as they are usually not sufficiently long. They are composed of printable characters, thus they do not meet the requirement of being uniformly distributed. If a human should be able to remember them, they will probably contain dictionary words, which increases their discoverability even more. Running a password through a PBKDF derives one or more cryptographic keys from it. These derived keys are pseudorandom and sufficiently long to make brute-force guessing as time consuming as possible.

Lately PBKDFs are taking on another defensive role. Due to the availability of GPUs, FPGAs, and ASICs, there are new possibilities for running functions in parallel in computing environments, increasing the effectiveness of brute-force attacks. PBKDFs try to defend against such attacks by using memory-hard algorithms to slow down potential attackers and to make running the function in parallel extremely expensive or inefficient. To point out one example, cracking

an eight-character passphrase used to unlock an encrypted volume in around two seconds on a Raspberry PI could take up to 1,085 NVIDIA Tesla P100 GPUs, costing about 120 million dollars. Trying to crack a volume encrypted with Argon2 created on a modern laptop would require up to 75,121 powerful machines running for ten years and cost over 4 billion dollars.

WHEN THE BACKUP SYSTEM NEEDS A BACKUP SYSTEM

It did not take long for password crackers to get the better of PBKDF2 by finding weaknesses in their parallel computing and GPU optimization. The usual way of attacking is to use brute force to try as many variations of keys as possible, or to use a dictionary approach that assumes passwords are based on actual words. Memory-hard functions such as Argon2 were a mechanism for making the compute power required for these attacks simply too expensive or time consuming for the efforts to be worthwhile, except in the rarest of cases.

Even the protections offered by Argon2 were soon not enough, as a 2016 paper concludes (https://eprint.iacr.org/2016/759.pdf). Making multiple passes through the hashing algorithm became essential to protect a password: where 6 passes had at first been considered "paranoid," 10 passes were now recommended.

PRICING THE COST OF A HACK

Based on previous assumptions of an attacker's options, I created a price model that would estimate costs connected with finding the right passphrase to unlock a LUKS2 encrypted volume. These costs include purchase of devices and electricity costs. The variables shown in **Figure 1** were defined to formalize the model.

The model expects that an attacker uses an array of homogeneous machines and that the speed of Argon2 hash cracking is at least approximately benchmarked. It does not assume any hints about the password. It expects that passwords are distributed uniformly through the password space and therefore an attacker should on average search through one half of P to recover the password.

The model can simulate two different cases. In the first case, an attacker plans to purchase actual physical hardware, and, in the second case, an attacker rents machines from an online cloud provider (Amazon, Microsoft Azure, Alibaba Cloud, etc.). The final price is determined by the equations shown in **Figure 2**.

$$N = \frac{\frac{S}{60 \times 60 \times 24} \times \frac{P}{2}}{L} \tag{5.1}$$

$$F = (N \times H) + (N \times D \times E \times 24 \times L) \tag{5.2}$$

$$F = N \times R \times 24 \times L \tag{5.3}$$

***Figure 2**. Equations for determining the cost of password discovery*

Equation 5.1 determines how many machines are needed to exhaust the complete password space P in L days. The result denoted as N can then be

D – power draw of one machine expressed in kilowatts

E – price of electricity expressed in dollars per kilowatt hour

F – expected final price of whole attack expressed in dollars

H – initial price of one machine (CPU, RAM, accessories) expressed in dollars

L – expected length of an attack expressed in days

N – number of machines expressed as an integer

P – number of passwords contained in the chosen password space expressed as an integer

R – price to rent one machine for one hour expressed in dollars

S – speed of one machine expressed as number of seconds spent computing one hash

***Figure 1.** Variables in the password discover-cost formula*

used in equations 5.2 or 5.3. Equation 5.2 is used in the case of purchasing and using physical hardware. In that case, an attacker could estimate the power draw of a single machine by collecting information from data sheets or by performing real world tests. There exist online versions of power consumption calculators. If an attacker decides to allocate computing resources in online clouds, then equation 5.3 should be used. Online cloud providers usually provide online price lists for their services.

> ...if these are the calculations that an attacker might make, what calculations should a defender be making?

THE PATH FORWARD

Where could this research lead next? The main method for simulating an attack against Argon2 was based on the Argon2-gpu-bench benchmarking tool. Only the computation of Argon2 is involved; no actual decryption of volumes is performed. It would be interesting to create a fully working cracking tool and test its performance, eventually making the price model more exact. This process could be connected with measuring the real power draw of physical machines while performing the attack. These values would make the model even more precise.

Another area for further research is using cloud-based computing services. The prices and estimates described in my thesis were made only based on theoretical information, and no actual computation was made using these resources. The real efficiency might differ, and in that case it might positively or negatively influence the resulting price and suitability for the attack.

One might wonder, if these are the calculations that an attacker might make, what calculations should a defender be making? RH RQ

ACKNOWLEDGEMENTS

I would like to acknowledge the following support:

Computational resources were supplied by the Ministry of Education, Youth, and Sports of the Czech Republic under the Projects CESNET (Project No. LM2015042) and CERIT-Scientific Cloud (Project No. LM2015085) provided within the program Projects of Large Research, Development, and Innovations Infrastructures.

Additionally, thanks go to my advisor Milan Brož for very inspirational supervision and for help with gaining access to hardware needed for benchmarking. Furthermore, I would like to thank Ondrej Mosnáček for his consultations concerning Argon2 and for the software used in this thesis. The English version of the complete thesis is available at https://is.muni.cz/th/yinya/?lang=en.

RESEARCH
QUARTERLY

VOLUME 2:2

Red Hat

Feature

Don't blame the developers: making security usable for IT professionals

Historically, usability studies have looked mostly at end users, doing focus groups or user testing with customers or the general public. This process often neglected developers, system administrators, and other IT professionals and the systems they use day to day. Our research focuses on the usability of Transport Layer Security (TLS)—specifically, handling the X.509 certificates—for IT professionals, investigating library APIs, command-line interfaces, manuals, and documentation. Cooperating with developers of these tools, we aim to make them more secure through better usability.

by Martin Ukrop

The vast majority of usability studies of security focus on end users who lack extensive IT experience. They revolve mostly around passwords or other forms of authentication, mental models of security, mobile app permissions, or browser warnings. When looking at the headcount, these users do form the majority of the user base, by far. However, the impact of their security mishaps usually involves only themselves. System administrators and support engineers, although much smaller in number, have a much greater influence. If they err, tens or hundreds of end users are usually affected. The impact gets even higher when we look at endpoint software developers, and higher still with the library or OS developers' decisions (and possible failures), which influence millions. Security being usable for them is, therefore, of utmost importance.

> Lousy usability for developers can have serious security consequences.

AN EXAMPLE OF UNUSABLE SECURITY

Let's look at the API of cURL, the ubiquitous library for transferring data with URLs. When initiating a secure connection with a server, cURL needs to verify the server's authenticity. This is most commonly done by validating the server's X.509 certificate. In the cURL API, two flags are controlling the certificate validation process:

'CURL_SSL_VERIFYPEER' configuring if the certificate should be validated at all,

About the Author
Martin Ukrop is a PhD candidate at the Centre for Research on Cryptography and Security at Masaryk University in the Czech Republic. His research in usable security is supported by Red Hat Czech. He is one of the founders of the Teaching Lab, a faculty platform supporting novice computer science teachers.

Team of the research booth at DevConf 2020 in Brno (from l. to r.): Lydia Kraus, Pavol Žáčik, Machaela Balážová, Martin Ukrop

and 'CURL_SSL_VERIFYHOST' specifying in what way to compare the certificate subject name with the server hostname. The devil, however, hides in the details.

Consider the PayPal Payments SDK, a major worldwide online payment system. In early 2012, the production code had a bug and—incorrectly and very insecurely—set both 'CURL_SSL_VERIFYPEER' and 'CURL_SSL_VERIFYHOST' to 'FALSE'. This meant that secure connections from PayPal SDK using cURL were not checking the server's identity, opening the door to many attack vectors. Fortunately, the bug was spotted and "fixed": on April 27, 2012, the developers set both these cURL flags to 'TRUE'.

Where is the problem then? Well, while 'CURL_SSL_VERIFYPEER' is indeed a boolean, 'CURL_SSL_VERIFYHOST' is an integer, where the setting of zero disables the hostname verification, the setting of one is a non-enforcing debug option, and the setting of two enables the full hostname verification. And in cURL, since it's written in C, the value of 'TRUE' is implicitly converted to one, effectively disabling hostname verification and allowing connections to servers with valid, but possibly stolen, certificates.[1]

Who is to blame? Is it the PayPal developers making a mistake? They were definitely not alone: at the time, similar bugs were present in ZenCart, Amazon Flexible Payments, Apache HttpClient, and Trillian. Is it the cURL developers for the inconsistent (and slightly counterintuitive) interface? The documentation clearly stated that the flags work this way. Is it the designers of the language of C for allowing silent coercion of variables? In fact, probably all of them share a bit of responsibility. Nevertheless, this and other similar examples have shown the developer world the extent of security consequences that can be caused by lousy usability for developers.

THE WORLD OF CERTIFICATE VALIDATION

Our usability security research revolves around X.509 certificates, their generation, validation, and understanding. Why? Nowadays, most developers need secure network connections somewhere in their products. Today, that mostly means using TLS, which, in turn, most likely means validating the authenticity of the server by validating its certificate.

Furthermore, it turns out that understanding all the various quirks and corners of certificate validation is far from straightforward. OpenSSL, one of the most widely used libraries for TLS, has almost 80 distinct error states related only to certificate validation. Managing such an error

[1] Many other examples of usability flaws of SSL APIs can be found in the article "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software," by M. Georgiev et al., published in the *Proceedings of the 2012 ACM Conference on Computer and Communications Security.*

landscape gets complicated and thus not all the errors convey the explanation and security consequences well enough.

Proper understanding of errors is, however, essential. Imagine you are attempting a TLS connection and the certificate validation fails with the code of 'X509_V_ERR_PERMITTED_VIOLATION'. You look it up in the documentation only to learn that "the permitted subtree was violated." If you wave it off as something unimportant, you risk connecting to a malicious server. What does the error mean? The issuing CA was constrained to issue certificates only for a given (sub-)namespace ("subtree"), and this particular one violates the restriction. Thus, the error may even indicate suspicious activity at the CA level!

Of course, you might claim that developers would not continue connecting with a certificate error. In that, however, you would be wrong. Multiple studies (including our own, described below) show that users, including developers, routinely bypass certificate warnings and errors if it's possible. And to determine which errors will offer the user the possibility of clickthrough and which will not, the developers need to understand the errors in the first place.

USABILITY OF CERTIFICATE HANDLING TOOLS

Our first study took place in 2017, among the developers attending DevConf.CZ, an open source community conference in Brno, Czech Republic, organized by Red Hat. We set up a booth and asked developers, administrators, and other IT professionals passing by to generate and validate a handful of certificates using command-line OpenSSL. While on it, we were watching where they struggled and what resources they used.

The usability of OpenSSL turned out to be far from ideal. This is supported by the participants' subjective opinions—many avidly said they hate interacting with OpenSSL—as well as the objective measures of the task results. For example, 44% of the participants were unsuccessful in generating a self-signed certificate, while thinking they had succeeded. In the validation task, 71% of the participants misconfigured or omitted the inclusion of the root trust store of the operating system, even though they were explicitly instructed to consider those roots as trusted.

Documentation, such as manuals, tutorials, or Q&A forums, appears to matter a lot. The majority of participants used both online sources and manual pages to solve the task. The Stack Overflow forums were a repeatedly used resource (73% of our participants used it), but it's not the only one. It seems that any well-written tutorial can be widely used: in our task, the most visited tutorial page was a semi-random page in the knowledge base of the University of Wisconsin (40% of the participants), just because it covered one of the tasks well and scored high in search results. The importance of tutorials becomes even more prominent when we realize that even developers

Martin Ukrop interviews developers at DevConf 2018, Brno, Czechia.

tend to copy-paste the suggested commands without further adjustments (in our study, only 9% of the participants altered the copy-pasted command).[2]

UNDERSTANDING AND TRUSTING CERTIFICATES

Our follow-up experiment at DevConf.CZ 2018 investigated how much developers trust flawed TLS certificates. Participants were put in a scenario of improving the conference website to allow registration using federated identities. However, the connection to authentication servers failed with certificate validation errors. We then asked the participants to investigate the issue, assess the connection's trustworthiness on a given scale, and describe the problem in their own words.

The results clearly show that trust decisions are not binary. Even IT professionals do not entirely refuse a certificate just because its validation check fails. In the case of an expired certificate, the expiry duration plays an important role: certificates expired yesterday were mostly considered as "looking OK." In contrast, a certificate expired two weeks ago "looks suspicious," and the one expired a year ago seems "outright untrustworthy." Certificates of different subjects were regarded differently: flaws were less likely to be tolerated for big, established companies.

Even more importantly, some certificate cases were overtrusted. For example, 21% of the participants considered the self-signed certificate as "looking OK" or better, and 20% saw the certificate with violated name constraints as "looking OK" or better. The mean trust in both cases was comparable to that of an expired certificate. We find this quite concerning: the self-signed certificate does not have any identity assurances (literally anyone could have created it), and name constraints violation hints at misconfiguration or even malicious activity at the subauthority level.

> Improving the usability for developers can result in making the tools more secure.

In the spirit of positive change, we were curious to find if better error messages and documentation would improve understanding and trust perception. For our next step, half of the participants interacted with the real OpenSSL errors and the other half with our redesigned version. Seeing our reworded errors and documentation, both self-signed and name-constrained cases seemed significantly less trustworthy and required less time and less online browsing to understand. These results confirm one more time that usable documentation is a crucial part of software design.

When investigating how to direct programmers to a useful documentation source right from the error, we experimentally included a documentation URL directly into the CLI error message. To our surprise, 71% of the participants clicked this link. Unusual as it is, it suggests a viable way of directing developers to a helpful resource recommended by the library designers.[3]

USABLE ERRORS AND DOCUMENTATION

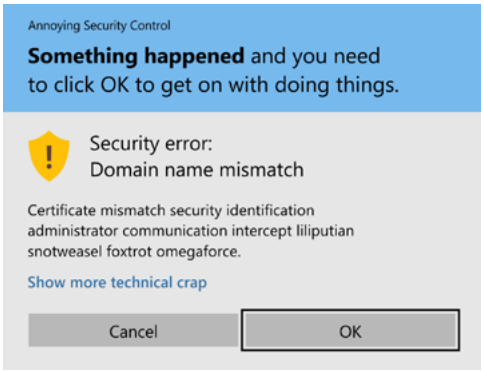
In light of our research results, we decided to make certificate validation errors and corresponding documentation more usable. Currently, there are many different libraries used for handling TLS connections and validating certificates. Plurality is welcome, but the differences in these tools complicate knowledge transfer and transitioning the project from one library to another. In the long term, we aim to simplify and unify the ecosystem by standardizing the validation errors and providing reliable developer-tested documentation. Our work in progress is already available at https://x509errors.org.

First, we are mapping the landscape of certificate validation errors in multiple libraries, starting with OpenSSL (openssl.org), GnuTLS (gnutls.org), Botan (botan.randombit.

[2] Research from this section was published in the paper "Why Johnny the Developer Can't Work with Public Key Certificates" at the RSA Conference 2018 Cryptographer's track.

[3] Research from this section was published in the paper "Will You Trust This TLS Certificate? Perceptions of People Working in IT" at the Annual Computer Security Applications Conference (ACSAC) in 2019.

net), and mbedTLS (tls.mbed.org). Their errors vary vastly in number, granularity, and documentation. To ease debugging for software developers, we started generating and publishing example certificates exhibiting every individual error. As of now, we have 34 errors covered by automatically generated certificates for public use in software development.



How many end users and developers actually understand error messages of X.509 certificate validation? (joke adapted from Johnathan Nightingale)

Second, we are trying to identify the corresponding errors in different libraries. For example, a certificate with the aforementioned OpenSSL error 'X509_V_ERR_PERMITTED_VIOLATION' will get a 'CERT_SIGNER_CONSTRAINTS_FAILURE' in GnuTLS and a rather general 'X509_BADCERT_NOT_TRUSTED' in mbedTLS.

Third, seeing all the errors and the corresponding pieces of documentation in one place will enable us to design a unified taxonomy of certificate flaws. To be able to add reliable documentation to this taxonomy, an active discussion with developers is needed. In January 2020, we conducted another study with participants of DevConf to look into the matter. We designed alternative documentation for three errors and asked the IT professionals for feedback regarding its content and structure. After we finish analyzing the results, we aim to propose a draft of the new documentation.

MAKING THE WORLD A BIT MORE USABLE

Usability is, in general, difficult to achieve in systems as complex as TLS. Furthermore, the upstream changes are complicated by the need for compatibility preservation. Nevertheless, we propose at least the smaller, ready-to-adopt changes. In cooperation with the OpenSSL developer community, we have already got upstream several patches regarding documentation and plan for more in the future.

Acknowledgements:
Thanks are due to Vashek Matyáš for supervision, Pavel Žáčik for his dedicated work, Red Hatters Jan Pazdziora and Nikos Mavrogiannopoulos for the initial support, and Milan Brož for long-term facilitation of the academic cooperation of Red Hat Czech and the Faculty of Informatics at Masaryk University.

More detailed results of this research project can be found on the Red Hat Research website at https://red.ht/3dxBnpa.

Trust is not binary. Not even for IT professionals.

Feature

Isn't multi-tenancy Ironic?

Virtualization is an amazing technology that has become a popular solution for sharing resources among members of an organization. However, some organizations need to harness the capabilities of an entire machine, without a layer of virtualization between the code and the hardware. Is it possible to share hardware between projects with the same ease as sharing virtual resources?

by Tzu-Mainn Chen and Lars Kellogg-Stedman

About the Author

Tzu-Mainn Chen is a Principal Software Engineer at Red Hat, currently focusing on bringing Elastic Secure Infrastructure to life for the Massachusetts Open Cloud.

About the Author

Lars Kellogg-Stedman is part of the Office of the CTO, where he works on Red Hat's collaboration with Boston University and the Massachusetts Open Cloud.

Research into this question made clear that the flexibility and security necessary for sharing hardware created a host of additional complexities. The Massachusetts Open Cloud created an initiative to simplify those complexities, with a goal of enabling a multi-tenant, bare metal cloud where users can lease bare metal servers and create private networks to form their own isolated bare metal clusters. Initial work centered around the HIL (open.bu.edu/handle/2144/19198) and BMI/M2 (ieeexplore.ieee.org/abstract/document/8360313) projects. Those efforts have morphed into the Elastic Secure Infrastructure (ESI) project, which will build a solution on top of an established open source cloud platform: OpenStack (openstack.org).

The OpenStack service that manages bare metal is called Ironic. When we looked at Ironic, we discovered that it included many features we needed. However, it was missing a key element: bare metal node multi-tenancy. What does that mean? While most OpenStack services allow each project to own their own resources, Ironic provides an all-or-nothing view of bare metal hardware: either you're an administrator and have access to everything, or you're not and see nothing.

A FRAMEWORK FOR IRONIC MULTI-TENANCY

Node multi-tenancy has long been a nice-to-have goal for Ironic: a feature acknowledged to be useful, but which slips behind other important requirements in release after release. Fortunately OpenStack is an open source project, so we approached the upstream Ironic community and asked them how we might roll up our sleeves and implement the feature ourselves. They helped us document a plan that consisted of the following steps.

1. Make the Owner field operationally significant.

Ironic added an Owner field for nodes back in 2018, but it was strictly for informational purposes.

The first step in bringing multi-tenancy to Ironic was to allow the Owner field to be used for access control.

2. Add a Lessee field to Ironic nodes. Many of the workflows we envision involve a hardware owner temporarily lending their hardware to another project. This requires a mechanism for recording this secondary operator: the lessee.

3. Create new policy rules for owners and lessees. Like most OpenStack services, access control in Ironic is managed through policy rules attached to various actions. Adding policy rules related to node owners and lessees allows administrators to grant node API access to non-admins.

These updates allow each Ironic node to specify a project as an owner (a project that will always be able to control the hardware) and/or a lessee (a project that will be able to control the hardware for the duration of a lease). An Ironic administrator can then expose individual node API calls to node owners and lessees by updating a policy file. For example, you can configure Ironic policy to allow:

- owners and lessees to control an individual node's power state
- owners (and not lessees) the ability to modify node attributes

These changes provide the foundation for a shared bare metal cluster, but they aren't a complete solution. For example, provisioning an Ironic node requires the ability to update *some* node attributes, but granting a lessee the ability to update *any* node attribute would allow them to steal control of the node from the owner. What's the solution? Simple: we created new policy rules with granular access control for specific node attributes. Lessees can then have restricted update access to these attributes, while owners still enjoy the ability to modify any node attribute they want.

The upstream Ironic community also pointed out additional Ironic resources that would be affected by node multi-tenancy. With their guidance, we added multi-tenant support for managing network ports and allocations (objects used for scheduling bare metal nodes when operating Ironic in standalone mode).

After all this was done, we attempted to provision a bare metal node using Metalsmith (metalsmith.readthedocs.io/en/latest), an upstream tool created expressly for this purpose, to see what further gaps we might have to close in order for non-admins to use the tool successfully. To our surprise and gratification, Metalsmith worked seamlessly with our changes, with no additional work necessary—a pretty good validation of the implementation!

WHAT'S NEXT FOR ESI?

The code for all this multi-tenant functionality is done and merged, and should be part of the OpenStack Ussuri release (opendev.org/openstack/ironic). Furthermore, during the 2020 OpenCloud Workshop we held a positive Birds of a Feather discussion (research.redhat.com/wp-content/uploads/2020/05/ESI_Ironic-Presentation.pdf) on multi-tenant Ironic with both ESI and OpenStack contributors participating. What comes next? We're starting research into the networking and storage needs of ESI. We're also planning a new leasing service that sits on top of all of this: a service that makes it easy for owners to offer up nodes for lease, and for lessees to claim those nodes.

Additional initiatives promise to expand the scope of ESI in exciting ways. Last summer, Boston University students collaborated on FLOCX (youtube.com/watch?v=goDpCRLhCao): a marketplace for bare metal with an early proof of concept that was demonstrated at DevConf.US 2019. Other researchers are investigating increased bare metal security through non-intrusive software introspection (research.redhat.com/wp-content/uploads/2020/04/Mohan_NISI.pdf) and secure allocation of resources (usenix.org/system/files/conference/hotcloud18/hotcloud18-paper-mosayyebzadeh.pdf).

ESI is a complex project with a lot of moving parts, but we're excited to see it start to come together! If you'd like more information, browse through our Git repository at github.com/CCI-MOC/esi, or join the #moc IRC channel over on FreeNode.

Interview

Voyage into the open Dataverse

The next frontier in balancing data sharing needs with privacy protection

by Sherard Griffin

About the Author

Sherard Griffin *is a Director at Red Hat in the AI Center of Excellence. His primary responsibility is the development of Open Data Hub, a community-driven reference architecture for building an AI-as-a-service platform on Red Hat® OpenShift®. He is also responsible for the deployment of Open Data Hub, which processes hundreds of gigabytes of data per day. The stored results are made available to analysts, developers, and data scientists across the company.*

*We spoke about the importance of data sharing and privacy preservation, in both scientific and computer technology domains, with **James Honaker** and **Mercè Crosas**, two of Harvard's leaders in these fields. They discussed how we can make open source solutions for storing and sharing richly detailed information about experiments, software, and systems more available to all.*

Sherard Griffin: James and Mercè, I'd love to hear more about what you do at Harvard and the projects you're associated with.

James Honaker: I work with a team in the computer science department doing research on adjacent computing. We take algorithms being developed by researchers and try to build prototypes from them. Or we take prototypes we've developed and try to turn them into more robust user tools—basically try to get code out of theory and into tools.

One we've focused on a lot has been building systems for privacy preservation: using the mathematical theories, definitions, and algorithms of differential privacy in a library- and researcher-oriented way, so pragmatic researchers, data scientists, and statisticians can leverage differential privacy without having any expertise.

Mercè Crosas: I have two roles at Harvard. One is a university-wide role, with the Senior Leadership Team of Harvard IT. The university has a data management office, so I work with the CIO to help organize how we use data across Harvard. The other role is within the Institute for Quantitative Social Science (IQSS), as a senior science and technology officer. I've been with the Institute for about 15 years, and within this time we've built a public platform to build data repositories called Dataverse.

Dataverse has an open source community around it and a couple of research projects associated with it. More recently, several projects within IQSS work on improving research by building research tools or providing data science consulting and training services. We opened the Project for Differential Privacy in the context of how data repositories will get integrated with data privacy solutions.

Sherard: James, you mentioned that you focus a lot on building out systems and tooling for data privacy. What sparked your interest in data privacy?

James: It started when I was at IQSS with Mercè. The CS theorists, people who work on

Two of Harvard University's leaders in data sharing and privacy preservation in scientific and computer technology, Mercè Crosas (left) and James Honaker (right), talk about open source solutions with Red Hat's Sherard Griffin.

proving mathematically what's possible with computation, had this large body of literature proving what things were possible or impossible in terms of privacy. IQSS was partnering with them to work out if any of these could be used as a tool for applied researchers. Somehow I got stuck in the bridging role for a little while, which was, "people don't know how to talk to these people, so why don't you go be the person who talks to them?"

Sherard: That's certainly a valued asset.

James: Yes, I think it means I had grasped just enough content in enough fields to play translator. As it turned out, some of that bridging

work involved communicating with scholars about what people did with data and how some of the things we already understood in statistics could translate. I ended up moving to their institute so I could do more work with them.

Red Hat's Sherard Griffin (top right) and Heidi Dempsey (bottom right) sit down for a virtual chat with James Honaker (bottom left) and Mercè Crosas (top left).

Sherard: A lot of what we do at Red Hat and the office of the CTO is similar. We try to bridge the vision of what a product is working on versus what customers need. I'm curious how that gets incorporated with what companies like Red Hat and the open source communities do. What does that look like from your end, where you're taking something theoretical and making a repeatable tool?

James: Getting code that works within the pragmatics of how computing actually occurs is something we push on. For example, these proofs are often written in the space of real numbers, but computers have to use floating points, and that difference tampers with the proofs. Or sometimes, things are sort of order of magnitudes and they don't care about the exact constants. Those get wiped out of the proof, but if you can change the utility of an algorithm by a factor of two or ten, that matters a lot to a researcher, whether they have half their data or a tenth of their data ending up usable.

We also work with use case partners. When you talk with an analyst who's got a very specific use case, they have private data from one government entity, private data from another government entity, and maybe it has some weird distribution. And oh, this is the very hard way that we have to join them. All of those pragmatics end up pointing out things that weren't quite covered by the original theory.

Sherard: Does that feedback work its way back to the researchers? I imagine the real world use cases are far different than what you would see in an isolated research environment.

James: The feedback loop is definitely a nice part of the iteration. It informs research agendas. Sometimes we pitch in on that. Sometimes we have ideas and then somebody will go off and prove that our idea actually makes sense. Sometimes, we can point out, "Look, your algorithm will be a lot more useful if you could do X, Y, and Z." Then they go and work on that.

Sherard: Mercè, tell us about how you got into data security and the privacy side of your work.

Mercè: So, part of the data works project we started at IQSS was making the data available as openly as possible. In the last years, there's been a move to more open science and data. But there are also requirements from funding organizations. If you publish in a journal, you have to make the dataset you used publicly available. The problem is, sometimes research uses sensitive data containing information about individuals. If we cannot make it available, we cannot reuse datasets to reproduce results that have been published.

So, how could we find something in between? How could we build a platform from open data, using even the most sensitive dataset, in a way that organizations would still allow us to access some of the summary statistics or some of the results of their dataset? It was a practical necessity to get involved and provide solutions for accessing sensitive data for research.

Sherard: For those not familiar with Dataverse, can you give a couple sentences on what it is and how we can tie it back to differential privacy?

Mercè: Yes. Dataverse is a software platform enabling us to build a real data repository to share research datasets. The emphasis is on publishing datasets associated with research that is already published. Another use of the platform is to create datasets that could be useful for research and making them available more openly to our research communities.

Sherard: One of the challenges we've faced at Red Hat is that the datasets we needed from a partner to create certain machine learning models had to have a fair amount of information. Unfortunately, the vendor had challenges sharing that data, because it had sensitive information in it. Have you run into scenarios where you're trying to do analysis or machine learning on this kind of data? How would differential privacy or OpenDP help out?

James: That's a great use case. So, differential privacy is a mathematical definition, not an algorithm. An algorithm either meets the definition or it doesn't. If an algorithm is proven to meet that definition, you can reason about the use of that algorithm formally and make guarantees. Loosely speaking, the guarantee is that the releases, query answers, or models that your differentially private algorithm provides won't leak information about any one individual. They can't even learn whether or not I was in the dataset in the first place or get a distribution of answers affected by my information. It's a very, very high, gold-standard guarantee.

> In the last years, there's been a move to more open science and data. But there are also requirements from funding organizations.
> —*Mercè Crosas*

Normally, these algorithms are adding a small amount of noise sufficient to drown out the contribution of any one individual in the dataset. So you don't have to strip out all of these potentially sensitive attributes, because there's no way to attach those to any individual. Stripping out sensitive data makes analysis really hard to run. Maybe the relationship between the sensitive variable and some other characteristic you care about is the fundamental quantity of interest. You strip out the sensitive data, you can't do anything.

Sherard: It sounds like it would be quite a challenge to know how far to obscure the data or how much noise to add to make sure you don't add too much.

James: You cut exactly to the heart of the question. I call it very fine-tuned lying. If the noise is too great, you're losing more utility than you needed. If the noise is too low, the privacy guarantee goes away. The point is to balance that noise exactly; that's why the ability to reason formally about these algorithms is so important. There's a tuning parameter called Epsilon. If an adversary, for example, has infinite computational power, knows algorithmic tricks that haven't even been discovered yet, Epsilon tells you the worst case leakage of information from a query. So what you decide is, "Okay, how far do I think I am from that worst case? How much information would I be willing to give such an attacker in order to release a query?" That tells you what that noise has to be.

Sherard: I want to come back to reproducibility. In software, we try not to release without having some level of continuous integration testing and ability to validate that the application will behave a certain way given certain parameters. What does that mean in the world of data science?

Mercè: Good question, and not an easy one. Computational reproducibility is a big topic. When I refer to reproducibility, I'm talking

about using the same data and the same methods to see if you end up with the same results. So it's validating the work, the scientific outcome of your research, which is an eternal battle.

Sherard: I imagine you would need to be able to share the data that created the results in the first place.

Mercè: Exactly. That's the connection with our work with Dataverse. A lot of the work connected to open science and open data is to make that possible. We also say the code and the software should be open, so you could reuse the same computation.

Sherard: How do you see the difference between differential privacy and OpenDP and some other privacy protecting technologies, like multiparty computing? Why is encryption not enough?

James: They're complementary. Most uses of encryption are about confidentiality. If I've got sensitive data, I don't want somebody to hack the system and do an end run around my interface and pull the data out or monitor it in transit.

But when I run an analysis on the data, I'm creating an answer I'm going to send out into the wild. I want to make sure that answer, after it leaves the system, can't be plugged into some attack that leaks information. That's what differential privacy is giving you. It's giving you an interface between computations you might want to run on the data and what you can publish in the outside world.

Once you start answering queries on a dataset, you are necessarily leaking information. Differential privacy ensures you never answer the questions too precisely or answer too many questions. Differential privacy is not encrypting the data. It's how you release things out of the system.

> It's a real barrier to entry that every single person has to start from scratch. So why don't we all build it together?
> *—James Honaker*

With multiparty computation, there are interesting connections. Multiparty computation is often how we share confidential data. How do we allow multiple people to have confidential data yet reach a common answer? Multiparty computation is one answer. There are differentially private ways of creating a differentially private answer. I create a differentially private answer, then we work out how to add the two together. That's another approach. There are researchers here at Harvard and at Boston University looking at the connections between the two.

Sherard: That's a very good answer. So, I'm thinking of these scientists whose timelines are decades, as far as how long they want to be reproducible. Our timelines in hardware and software are a couple of years. What can you do to help with that difference?

Mercè: We have a team looking into this problem and looking at ways, for example, of using Docker containers to encapsulate the environment, including code and data, where you run computational analysis and are able to reuse it. Of course, everything has a timeframe. The containers might even change. You need to look at it the way libraries and archives have done for decades to see how you solve the preservation problem. The best solution is one that's more preservable and least dependent on proprietary software. However, I don't think it's a problem that is solved for every case. Some things might be just too difficult to preserve long term.

One thing we're looking into is how to summarize the metadata that goes with the data so it's easier to rerun. Many times, the problem is the documentation. We're trying to see simple ways of summarizing this in a simple format for anybody to reuse.

Sherard: How can engineers collaborate on OpenDP and some of the differential privacy work you're doing?

James: What we've seen is that industry groups or tech companies build their own end-to-end, bespoke differential privacy system to solve one question they really care about, and they do it

really well. And lots of academic researchers build an end-to-end prototype that demonstrates one thing they've been researching, and they do that really well. There's a lot of overlap of cryptography, and the fundamental rule of cryptography is you never want to roll your own, right? You want everybody to be using the same underlying library, because then everybody else has vetted it.

It's a real barrier to entry that every single person has to start from scratch. So why don't we all build it together? Why don't we build one underlying library that everybody's contributing to, that's flexible enough that industry can use it for their problems, researchers can use it for their own cutting-edge directions.

Mercè did a lot of work building this OpenDP conference where people were discussing use cases like, "Here's what my data is, here's what analysts need to be able to do. This is what you need to be able to solve." And people were talking about systems engineering and saying, "Okay, this is how I put my data in the cloud, and this is how I need to be able to access it, and this is how it scales. Make sure it works." So there's lots of places for people to contribute their talents. The goal is to build a community of people who are willing to pitch in.

Mercè: We had a session on collaboration in the OpenDP workshop where we talked about the code library, the center of all this work, because you need that to build anything on top of it. But then, there is a whole layer of tooling that could be making use of that library to access the user interface, run queries. Then there is another layer of these end-to-end systems that says, well, let's say Red Hat wants to use data they don't have or the data within some of the tools to provide a system that includes differential privacy. Then we find ways to partner in building this.

Sherard: What challenges do you foresee OpenDP facing in the near future?

Mercè: One challenge to getting a differential privacy library is one many products face: getting this idea out there for people to use it. Most of the components are already there, we just need to release it in a way where we feel comfortable it can be transferred and verified.

James: I hope this is a sign of our mutual respect and adoration of each other, that Mercè sees the hardest thing as the thing I do—building the library—and I would say the hardest thing is what Mercè does. There are all these groups saying, "This is a great idea. I want it to work in my context." They're all pulling in slightly different ways. How do you build a community that's cooperative and balances all those interests? That seems like the phenomenally hard challenge.

Sherard: One last question. When do you think differential privacy will be used commonly in datacenters? Is this something we can achieve?

James: That's a good question.

So, I got involved in this project just as my daughter was being born. And the whole literature was very new. It was all about potential and theory and abstract things. Now my daughter's in kindergarten, and we've got actual systems that people are really using, and I think, "Okay, now the literature is sort of in kindergarten." I'm hoping by the time she gets to high school the literature will also be in high school, which is to say it'll know most of the subjects reasonably well and it'll be pretty well rounded. That's what Mercè and I are trying to push, but I hope it doesn't take ten years. I hope it's a very gifted child who gets to high school in five years.

The goal is to build a community of people who are willing to pitch in.
—James Honaker

Feature

Fostering open innovation in hardware

Why is open hardware important? How is the new RISC-V architecture bringing open hardware research to the forefront? How will this impact you? Read on to find out.

by Yan Fisher

About the Author

Yan Fisher is a Global evangelist in the Emerging Technologies team at Red Hat, where he extends his expertise in enterprise computing to emerging areas that Red Hat is exploring.

The demand for computational power continues to grow year over year, following the requirements imposed by the ever increasing number of applications and the need to process even larger amounts of data. At the same time, the approaching end of Moore's Law and Dennard scaling means that building the same types of central processing units (CPUs) with more dense structures (i.e., a greater number of transistors) on reduced node size (i.e., a smaller physical footprint) leads to diminishing performance gains, increased energy consumption, and, consequently, more stringent cooling requirements. We can no longer expect the traditional instruction set architectures (ISA) to keep pace with ever growing demand. The future of computing needs to be more heterogeneous in nature and, just like software today, it needs to become more open.

Red Hat is closely tracking and evaluating the potential impact of the forthcoming changes on traditional computing. We are focusing our attention on several key hardware categories and subsystems (https://research.redhat.com/wp-content/uploads/2019/12/RRQ-Vol2-1.pdf).

While computer designs are becoming more heterogeneous and the importance of CPUs has diminished, CPUs will be around for a long time, especially for popular architectures like x86. An apparent lack of ISA flexibility for CPU designs, combined with ongoing academic research around creating a royalty-free instruction set, presents a unique opportunity for an open and customizable ISA to emerge. RISC-V architecture takes that challenge head on.

A BRIEF HISTORY OF RISC-V

RISC-V is a free and open ISA that hardware designers can modify and experiment with. The unique aspect of RISC-V is that its design process and the specifications are truly open. The design reflects the community's decisions based on collective experience and research. Although the ISA is free, the processor implementation need not be: vendors can create commercial products without disclosing the underlying processor design. This is similar to the open source model for software, in that derivative work and modifications are allowed. Additionally, for any new design to use the RISC-V name or trademark, it has to maintain compatibility with the ISA.

While other architectures may claim to be open now, none of them were thought of as such at the onset or had an active ecosystem to support it going forward. This is similar to how

Linux® was open for development and contributions from the start, versus SystemV or BSD, which released their source code to the community years after they had been around.

The RISC-V instruction set was developed by Professor Krste Asanović and graduate students Yunsup Lee and Andrew Waterman in May 2010, as part of the Parallel Computing Laboratory at the University of California, Berkeley. In 2015, the RISC-V Foundation (riscv.org), a nonprofit corporation controlled by its members, was founded to build an open, collaborative community of software and hardware innovators based on the instruction set. In November 2018, the RISC-V Foundation announced a joint collaboration with the Linux Foundation (linuxfoundation.org)that provides operational, technical, and strategic support. Red Hat is a Silver member of both foundations.

In the last five years the adoption of RISC-V technology has been progressively growing on a global scale, predominantly due to the following factors:

- **Geo-political reach:** The openness of the RISC-V design makes it more likely to be trusted across the globe. The EU, China, and India all have ongoing projects that are supporting development and adoption of RISC-V processors.
- **Technical adaptability:** The flexibility to modify the ISA at the register and memory level—for example, adding machine learning or database-specific instructions—provides room for better code optimization. The net-new, clean-sheet design eliminates the requirements for supporting legacy instructions and backward compatibility while increasing flexibility and lowering the design complexity.
- **Accessibility:** The ability to prototype and extend designs based on RISC-V ISA as a learning tool leads to large-scale adoption in academia. This means that electrical engineers entering the workforce will have the skillset to build products based on the RISC-V ISA and will in turn promote the use of RISC-V.
- **Cost:** Unlike any other generally available ISA RISC-V's free-to-license model drives overall costs down, making it attractive to enterprises.

CURRENT AND FUTURE APPLICATIONS

The areas of application for this technology range from embedded microcontrollers to general-purpose and high performance servers. The potential is great, and RISC-V cores are already being used in embedded systems and IoT devices. In the next couple of years RISC-V will likely begin to move up into the server market.

Examples of RISC-V adoption range from announcements of support in the near future from Alibaba and European Processor Initiative (european-processor-initiative.eu/accelerator) to finished products from Western Digital, NVIDIA, SiFive, and Espressif.

- Western Digital has developed SweRV, a RISC-V processor for the controller that is part of the physical disk drive—a classic high-volume, low-cost embedded application. Western Digital's goal is to use RISC-V as the standard engine across their entire product line.
- NVIDIA is shipping millions of graphics processing units (GPUs) with an embedded RISC-V control processor that handles small but highly optimized tasks.
- SiFive currently offers two board-level products: HiFive Unleashed and HiFive1. SiFive products include two design tools, Core Designer and Chip Designer, as well as a set of RISC-V IP cores that are customized by the design tools.
- Espressif developed two SoCs, the ESP8266 and ESP32, that are extremely successful due to their low cost and many features. The most recent design, ESP32-S2, uses a RISC-V core as its ultra-low-power core.

The rate of RISC-V ISA adoption, as well as development of the RISC-V ecosystem overall, has benefited from the previous work done for other architectures using open source. For example, there is already a working and up-to-date OS and toolchain, which will only improve as the ecosystem focuses on optimizing them. That allowed RISC-V to skip traditional phases of software development and significantly lowered the barrier to entry to anyone interested in designing and using RISC-V softcores on FPGAs, or

even designing their own chips. While foundry costs are still significant and building a usable system around the ISA is still non-trivial, RISC-V is free and open for use by anyone, in all types of implementations, and remains unencumbered by licensing restrictions.

Similar to the early days of any new hardware architecture enablement, there is investigative work on RISC-V going on at Red Hat. It is currently an alternative architecture in Fedora, which boots and runs on several RISC-V emulators and boards. A team of community contributors is currently working on building the latest Fedora packages for RISC-V.

While RISC-V offers the ability to innovate freely and add unique capabilities in the form of ISA extensions, it is also intended to be a fixed ISA that adheres to industry standards and embraces standardization at the hardware level. These are both critical success factors for building an effective product and software ecosystem. In the future, customers might find RISC-V-based solutions attractive, since they would be highly customizable while supporting the standards-based software that makes up Red Hat's ecosystem.

Red Hat is encouraging the development of RISC-V in open source. We believe that open source hardware will naturally foster an ecosystem of open source software, and an open CPU architecture is a cornerstone of an open hardware platform.

RESEARCH
QUARTERLY

VOLUME 2:2

Red Hat

Feature

How to open source cloud operations

Open source has become a dominant paradigm for developing software. One major factor for its success is its transparency: if you have a problem with the software, you can peek into the details of the code, search the issue tracker, ask for help, and maybe even provide a fix. This means that even though most users don't write code, the mere fact that everything is open will help the majority of users. Now it's time to apply the open source model to the cloud.

by Marcel Hild

Today, in the age of cloud computing, we consume provided services that we expect to just work. And our applications are a complex mesh of those services. Developers need to configure software on demand with elasticity, resilience, security, and self-service in mind. That means the implementation and operations of those services, *i.e.,* the cloud, has become equally more complicated.

If open source made software great, how do we open source an implementation or the operation of something? By definition it's always different; there is no single binary that gets deployed multiple times. Instead it's an implementation of a procedure, a process. Same with operations: it's all the live data of metrics, logs, and tickets, and how software and the operations team react to it. So all implementations of a cloud, be it the large-scale proprietary public service or the on-premise private cloud, are snowflakes. Yes, best practices exist and there are excellent books. But still, you can't `` `git clone cloud` or `rpm -i cloud` ``.

If open source made software great, how do we open source an implementation or the operation of something?

EXTENDING ACCESS TO OPERATIONS

So we need to open up what it takes to stand up and operate a production-grade cloud. This must not only include architecture documents, installation, and configuration files, but all the data that is being produced in that procedure: metrics, logs, and tickets. You've probably heard the AI mantra that "data is the new gold" multiple times, and there is some deep truth about it. Software is no longer the differentiating factor: it's the data. Dashboards, post-mortems, chat logs– everything. Basically we need a public, read-only access.

No line of code gets merged if it does not pass the tests. But what about operations?

Signing up for read-write should be easy. Lowering the barrier of access was key to the success of open source, so let's lower the barrier to peek into the back office of the cloud as well. It opens up a slew of new opportunities. Suddenly we can create a real operations community. Current operations communities either center around a particular piece of technology, like the Prometheus monitoring community, or a certain approach to operations, like the Site Reliability Engineering (SRE) methodology. These are great, but we can also bring it down from the meta-level to the real world, where you can touch things. If you can't log into it, it does not exist.

We can also extend the community to people that operate their clouds. Those human DevOps people can watch and learn how a cloud is operated, then contribute by sharing their opinion on architectural decisions or their internal practices, and maybe even engage in operating bits of the open cloud. It's the same progression as in open source projects.

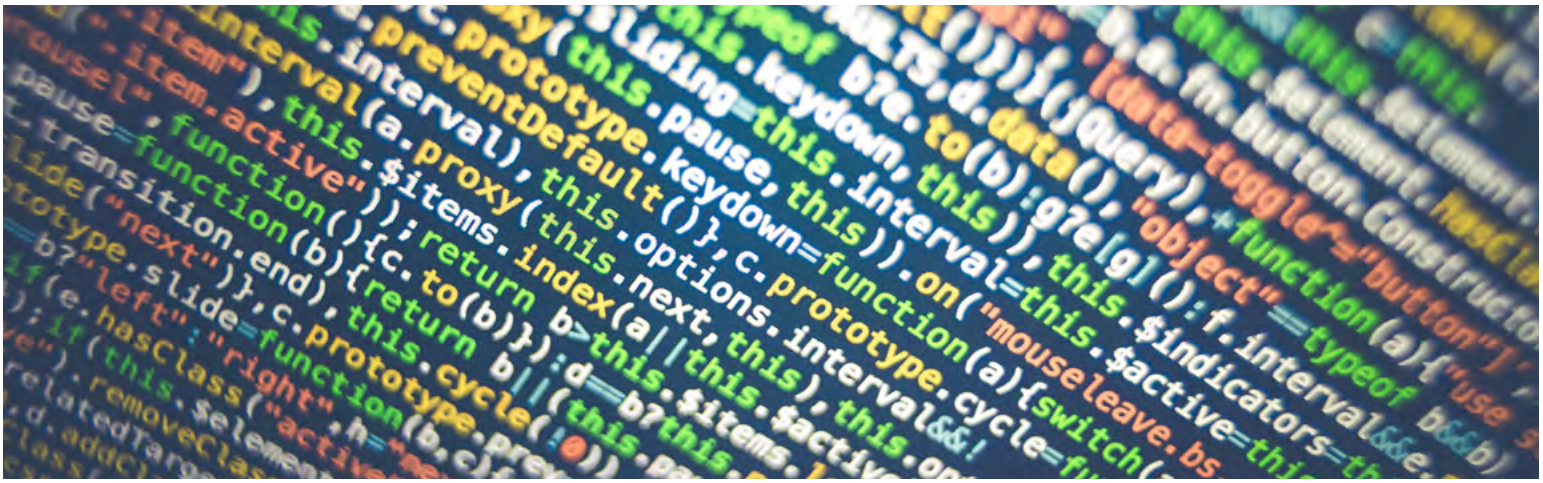
SHIFTING TO OPERATE FIRST

There's a principle in development called Shift Left, which means that we should involve testing really early in the development cycle—in other words, moving left in the process. This is already done with unit and integration tests. No line of code gets merged if it does not pass the tests. But what about operations?

At Red Hat we coined the term Operate First for this. The idea is similar to Upstream First, where we strive to get every line of code into an upstream project before we ship it in a product. In Operate First, we want to run the software in an operational context by the group that develops the software. And since we develop mainly in open source communities, this extends our open cloud to another group of people, the engineering community. The very authors of the code can be asked in an incident ticket about a misbehaving piece of the cloud. This not only increases the probability of getting the incident closed quickly, but it also exposes the software developer to the operational context of his brainchild. Maybe he comes back later and just watches how his software is being used and makes future design decisions based on the operations. The next level would be to try out new features in bleeding-edge alpha versions of a particular service and get a real workload instead of fake test data.

BRINGING IN AIOPS

Speaking of data, that brings us to the next audience of an open cloud: the research and AI community. AIOps is another term that is being used frequently—and to be honest it is as nebulous as the term cloud was a decade ago. To me, it means to augment IT operations with the tools of AI, which can happen on all levels, starting with data exploration. If a DevOps person uses a Jupyter notebook to cluster some metrics, I would call it an AIOps technique.



And since the data is available at the open cloud, it should be pretty easy.

But the road to the self-driving cluster is paved with a lot of data—labeled data. You will find large data sets with images that are labeled as a cat, but try to find data sets of clusters that are labeled with incidents. Creating such data sets and publishing them under an open license will spark the interest of AI researchers, because suddenly we can be more precise about a problem when we can be data driven. We can try to predict an outage before it happens.

Once the model is trained and tested against the test data, with the open cloud we can go even one step further. Researchers can collaborate with the operations team to validate their models against a live target. Operations can then adopt the model to enhance their operational excellence and finally involve software engineering. Ultimately, you want the model and the intelligence captured in code, right in the software that is being deployed—the software that will be deployed in another datacenter, in another incarnation of a cloud. That way, it will improve the operational excellence of all the clouds. This brings us closer to a world where operations of a cloud can be shared and can be installed, since it's embedded in the software itself. To get there, we need that feedback cycle and an open source community that involves all three parties—operations, engineering, and research—and we need a living environment to iterate upon.

Sounds like a story from the future? The process has already begun. Red Hat is working with an evolving open cloud community at the Massachusetts Open Cloud to help define an architecture of an open cloud environment where operability is paramount and data-driven tools can play a key role. All discussions happen in public meetings and, even better, are tracked in a Git repository, so we can involve all parties early in the process and trace back how we came to a certain decision. That's key, since the decision process is as important as the final outcome. All operational data will be accessible, and it will be easy to run a workload there and to get access to backend data.

If you're interested in collaborating, join us at openinfralabs.org.

About the Author

Marcel Hild has 25+ years of experience in open source business and development. He co-founded a Linux consulting company and worked as a freelance developer, a Solution Architect for Red Hat, and core Developer for Cloudforms, a Hybrid Cloud Management tool. Now he researches the topic of AIOps in the Office of the CTO at Red Hat, proving how AI will help operating machines and applications.

Project Updates

Greater Boston research update: June 2020

Faculty, PhD students, and Red Hat associates in the northeast United States are collaborating actively on research projects in many areas, despite the impact of COVID-19. The pursuit, testing, and examination of important research questions continues from spare bedrooms, kitchen tables, and even masked, socially distanced walks outside, with the support of many open source collaboration tools. Here we share recent highlights from some of our most active projects.

Researchers have been dispersed but not discouraged, as a recent story about Azer Bestavros (http://www.bu.edu/articles/2020/my-battle-with-covid-19-azer-bestavros/) and his battle with COVID-19 illustrates. A longtime Red Hat collaborator and Associate Provost for Computing and Data Sciences at Boston University, Bestavros eventually recovered and is looking ahead to future discoveries. We are starting to get involved in technology responses to COVID-19, such as the Private Automated Contact Tracing project (https://research.redhat.com/blog/research_project/pact-private-automated-contact-tracing/), which involves researchers from many universities working to preserve privacy while making exposure contact tracing faster and easier to do.

We are starting to get involved in technology responses to COVID-19, such as the Private Automated Contact Tracing project...

Students and faculty from many Red Hat collaborative research projects presented results and participated in panels at the 2020 Open Cloud Workshop (https://massopen.cloud/events/2020-open-cloud-workshop). We are starting more research projects related to the newly announced initiatives for the Open Cloud Testbed (https://massopen.cloud/connected-initiatives/open-cloud-testbed), New England Research Cloud, OpenInfra Labs (https://massopen.cloud/connected-initiatives/openinfra-labs-oilabs), and Operate First (https://massopen.cloud/connected-initiatives/operate-first). Check the workshop pages for full presentations and accompanying materials.

The Open Cloud FPGA Testbed initiative, led by Martin Herbordt, Boston University, and Miriam Leeser, Northeastern University, has been very active in projects on FPGAs in Large-Scale Computer Systems (https://research.redhat.com/blog/research_project/fpgas-in-large-scale-computer-systems/). A submitted paper on the first working multiparty computation FPGA example in a cloud instance depicts the use of FPGAs with secret sharing in the datacenter and examines the performance

improvement compared to a pure software implementation. Ahmed Sanaullah and Uli Drepper presented results on "Programming FPGAs the Open Source Way" (https://www.youtube.com/watch?v=P5gaw35L58I) at DevConf.CZ. Two PhD students from the University of Massachusetts are joining this group of BU and Red Hat collaborators during the summer to investigate open place and route tooling and DNN optimizations for FPGAs.

The Unikernal Linux (https://research.redhat.com/blog/research_project/unikernel-linux/) project continues to build on results presented last year at HotOS XVII, with several parallel projects. The April 2020 Eurosys paper "SEUSS: Skip Redundant Paths to Make Serverless Fast" (https://dl.acm.org/doi/abs/10.1145/3342195.3392698) showed how rapid deployment and high-density caching of serverless functions based on unikernel snapshots improved Function-as-a-Service platform throughput by 51 times on a workload composed entirely of new functions.

Researchers from the Elastic Secure Infrastructure project (https://research.redhat.com/blog/research_project/elastic-secure-infrastructure-2/) presented a paper entitled "Towards Non-Intrusive Software Introspection and Beyond" (https://research.redhat.com/wp-content/uploads/2020/04/Mohan_NISI.pdf) at the IEEE International Conference on Cloud Engineering in April. A related project to support multi-tenancy for bare metal machines for the open source Ironic project also completed and merged code that is included in the Usurri release. (See Tzu-Mainn Chen and Lars Kellogg-Stedman's article in this issue.) Multi-tenant Ironic slides (https://research.redhat.com/esi_ironic-presentation/) from an ESI Birds of a Feather discussion at the 2020 Open Cloud Workshop provide more information on the changes.

New PhD students from BU, UMass Lowell, and Worcester Polytechnic Institute are collaborating with Red Hat Research this summer. They will be contributing to open FPGA, open hybrid cloud, and AI projects. Expect to see updates on their new projects and progress on existing projects over the summer on research.redhat.com.

Check the complete project pages on research.redhat.com or contact academic@redhat.com for more information.

Partners

Automated Formal Verification

Honeywell and Red Hat have been collaborating with both the Faculty of Informatics from Masaryk University and the Faculty of Information Technology from Brno University of Technology on verification research for many years. These universities made Honeywell and Red Hat aware that they share the same business need: an automated detector of software defects. We joined our forces and started a three-year Automated Formal Verification project (AUFOVER) in 2019, co-funded by the Epsilon program from the Technology Agency of Czech Republic. Here, we share our results so far.

by Tomáš Kratochvíla

About the Author

Tomáš Kratochvíla is a scientist at Honeywell, where he automates verification and validation and brings the best benefits from the formal method to safety-critical systems. He leads the AUFOVER project and also creates requirement standards and methods for semantic requirement analyses.

Both Red Hat and Honeywell focus on detecting safety defects in C and C++ source codes, like arithmetic errors, illegal memory accesses, or control flow errors. While Red Hat is focused on huge code coverage and verification speed, Honeywell needs to make sure that its safety-critical software does not have even very rare bugs that are impossible to detect by commonly used unsound static analyzers.

All four AUFOVER partners are developing the following verification tools: Verification Server and Client Application, Scmock Plugins, DIVINE, and Symbiotic and Testos frameworks. While university tools DIVINE, Symbiotic, and Testos are based on formal mathematical methods, industrial tools encapsulate these into unifying distributed platforms that automate executions of the underlying tools. Honeywell tools are integrated based on Open Services for Lifecycle Collaboration specifications, which provides interoperability by defining how to integrate tools using linked data and REST API.

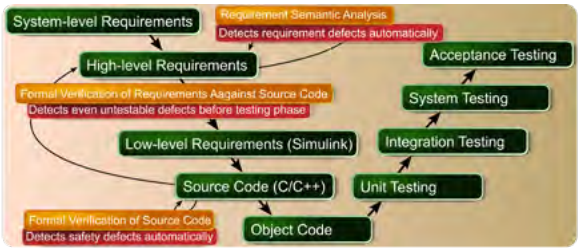


***Figure 1**. Typical development process for formal verification*

While formal verification tools are able to detect various defects, they are very difficult to apply for most developers—if they even know about their existence. Therefore, our approach is to offer

automated formal verification as a service and let the formal verification tools compete as to which can detect the most defects fastest and with highest confidence. Then the service aggregates all results and interprets the defects for the engineers. This approach is superior to running the complementary tools individually, since even the formal methods experts cannot predict which set of tools will be the most efficient and which tool parameters are the most optimal.

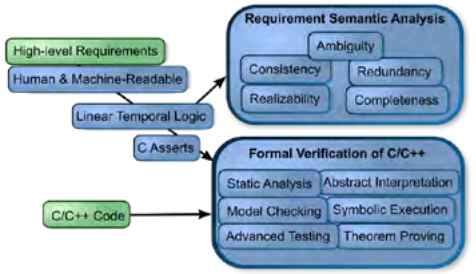


***Figure 2**. Automated formal verification workflow*

We focus on providing the following fully automated verification services:

Formal verification of source code: detection of defects that are failures in general, irrespective of any requirement, such as divisions by zero, buffer overflows, dereferences of null/dangling pointers, data races, or deadlocks. The problem with state of the art unsound static analyzers is that they report large numbers of false positives without guaranteeing that defects will be detected. Analyzing these potential defects is very time consuming. This is non-value-added activity for the developers—and they commonly introduce new errors during their trial to remove the false defect. This often creates distrust for these tools, so that developers may stop using them entirely.

Sound formal tools, on the other hand, are more difficult to automate and do not scale well. This is one of the reasons why running all the verification tools at once in parallel is a huge benefit. Formal tools will also automatically analyze potential defects from the unsound static analyzers and will remove false positives using witness checkers or witness validators. One possible approach is to run the potential defects through the Symbiotic tool, which slices the source code to relevant parts only and removes some false positives. This, in turn, increases the trustworthiness of the resulting verification report, while keeping the overall scalability of the formal verification.

Requirement semantic analysis: detection of defects in high-level behavioral requirements. Formal specification is very difficult for the engineers to create and very expensive to maintain on top of human-readable requirements, which are mandatory for safety-critical domains like aerospace. We have learned that requirement patterns and especially special requirement standards that allow engineers to naturally author requirements that are both human and machine readable are key enablers.

Ending up with formal specification and using the specialized formal methods tools is not enough. Our system also needs to interpret the defects to the user and explain why it is a problem, using specific examples. For example, when a set of requirements could be realized by a system that ignores some requirement conditions, an engineer may not understand why that is a problem from seeing an artificially generated transition system with thousands of states.

We can detect the following defects:

- **Ambiguity:** For example: "TrustVector1 or TrustVector2 has been less than 100 for 2 seconds" is an ambiguous part of a requirement. It is unclear if either condition should hold at any given time continuously for 2 seconds or if at least one condition should hold independently.
- **Inconsistency:** A set of requirements is logically consistent when no subset is contradictory for any evaluation of variables in time.
- **Redundancy:** A set of requirements implies another set, or some condition of a single requirement implies another.
- **Realizability:** Requirements are realizable by a non-trivial system and relatively complete when a system can be created that satisfies all requirements, does not restrict any input on top of the restrictions already introduced by the requirements, and no output could remain constant forever from the very beginning.
- **Missing requirements:** some behavior of a system is not constrained at all.

Formal verification of requirements against source code: verifies whether the source code satisfies the requirements optimally for any possible combination of input values in time. We automatically translate all the constrained formal requirements to Linear Temporal Logic, and we translate most requirements to transition logic with C Asserts that can be verified by most of the formal verification tools. Therefore, when any formal verification tool finds a combination of input values in time that result in assert failure, we can show this counterexample to an unsatisfied requirement or a part of a requirement to the user. Since we focus mostly on reactive systems, it makes sense to report to the engineer proof that the system satisfies given requirements for any possible execution of the system for at least a constrained time interval, since most of the complex systems properties cannot be completely proven.

We are benefiting from the already standardized API for the tools that compete in the Software Verification Competition (https://sv-comp.sosy-lab.org/) and similar competitions. In Software Verification Competition, Symbiotic and Predator (a part of Testos framework) tools consistently received the gold and silver medals for the last three years in the memory safety category. However, we also integrated several other tools, since some may outperform overall winners for specific systems. While advanced testing from Testos does not provide proofs and guarantees, it scales well and can find rare defects, for example using white noise insertion, other methods cannot, especially in parallel systems.

BENEFITING FROM COLLABORATION

In the first half year of this project, we evaluated these tools' performance on various types of industrial systems. Honeywell appreciates that universities are improving and customizing their tools to enable formal verification of highly complex industrial systems. Universities benefit from getting feedback on which of their methods to focus on and which problems are most significant in industrial systems.

We often found that we did not need to report a bug or a need for improvement in the formal tools since Red Hat had already done it a few days earlier. However, this is not the main benefit of our cooperation. We value Red Hat experience with both common static analyzers that scale very well and normalization of verification reports.

There is a huge potential for growth of our automated distributed verification system. We plan to extend it with improved scheduling of verification tasks, automated test case generators, security auditing software, and verification augmented with artificial intelligence. There is also a possibility to offer our verification service for other companies and users. Our engineers would like it if the verification also generated how to fix the defects and even automatically generated the system from the requirements. While this is possible for some simple systems, it could be computationally infeasible for even a few requirements.

At the end of the project, we will demonstrate the benefits of the integration of formal verification into our software development lifecycle. The benefits from discovering defects as early as possible during requirement authoring are appreciated by the engineers the most. However, it is difficult to compute the cost savings from this effect. The main benefits are expected on projects where the verification service can be deployed seamlessly using continuous integration; that is, whenever the requirements or source code changes in the repository, the verification report is automatically created.

DID YOU KNOW?

SAS® BRINGS ARTIFICIAL INTELLIGENCE AND ANALYTICS TO THE CLOUD.

You can run SAS on private, public or hybrid cloud infrastructures to better manage how AI work is done. SAS works with all major cloud providers to give you the power and freedom to innovate and be agile in the cloud.

sas.com/discover

sas