SECURITY AT EXTREME SCALES

Recommended Citation

Eric Grosse, "SECURITY AT EXTREME SCALES", NAPSNet Special Reports, May 30, 2019, https://nautilus.org/napsnet/napsnet-special-reports/security-at-extreme-scales/

ERIC GROSSE

MAY 30, 2019

I. INTRODUCTION

In this paper, Eric Grosse argues: “Much of the security progress over the past decade has been at large-scale, finding and patching vulnerabilities in widely used applications or defending networks of millions of machines containing high-value data. The lessons there may help military systems, but for the very highest security needs such as NC3, we ought to return to basics and harden small-scale systems. And we ought to do it as a joint effort, even between adversaries.”

Eric Grosse was Google's VP Security & Privacy Engineering. Before Google, Eric was a Research Director and Fellow at Bell Labs. He has a Ph.D. in Computer Science from Stanford.

Acknowledgments: The workshop was funded by the John D. and Catherine T. MacArthur Foundation.

This report is published simultaneously here by Technology for Global Security and **here** by Nautilus Institute and is published under a 4.0 International Creative Commons License the terms of which are found here.

The views expressed in this report do not necessarily reflect the official policy or position of the Nautilus Institute. Readers should note that Nautilus seeks a diversity of views and opinions on significant topics in order to identify common ground.

A podcast with Eric Grosse, Peter Hayes, and Philip Reiner on NC3 and security at extreme scales is found **here**.

The views expressed in this report do not necessarily reflect the official policy or position of the Nautilus Institute. Readers should note that Nautilus seeks a diversity of views and opinions on significant topics in order to identify common ground.

Banner image is by Lauren Hostetter of Heyhoss Design.

II. NAPSNET SPECIAL REPORT BY ERIC GROSSE

SECURITY AT EXTREME SCALES

MAY 30, 2019

Introduction

This is a personal perspective on information security at extreme scales, very large (billions of users) and very small (dozens of security-motivated users).

For "large-scale", the example I have in mind is Gmail, in which a diverse population world-wide uses the system to protect an even more diverse mix of valuable and frivolous assets. It is quite a challenge to build and operate something that meets their needs at the requisite convenience, cost, and ubiquity. Despite an astonishingly complex distributed infrastructure, a large investment in security has produced a mature defensive structure that is holding up reasonably well against advanced attackers. For "small scale" in the context of this workshop imagine an endpoint computer system used for communication within the larger world of nuclear command and control systems. The stakes are even higher, and some security lessons can carry over, but I argue that we need to take a much more conservative view of our ability to build verifiably secure systems. Complexity is the enemy of security.

Extreme Scales and Verification

When we say Gmail and associated products are extreme scale, perhaps the best published metric is the two billion[1] lines of code that run those applications and the web servers and the storage backend and the spam filtering and the network management and the development tools and the advertising auction and the tripwires and on and on. The physical realization of all this is distributed in offices and datacenters that need to be close to the users for good network latency and hence reside in geographic regions that may pose considerable security challenges.

Breaking into someone's email account may have considerable espionage value or enable substantial monetary theft and indeed talented global attackers are caught trying to break into Gmail. Despite this, to the best of my knowledge (which only covers recent years) there has been almost no loss of data from attacks on the infrastructure as opposed to phishing the user's password or getting a judge to sign a search warrant. That is a remarkable and unexpected achievement for a system of such complexity, and a tribute to heroic efforts by security researchers inside and outside Google. The normal arms race of global attackers and defenders seems to have been replaced with the friendlier race between security reviewers and fuzzers and blue teams and red teams, supported by a software engineering team that rushes to repair any cracks that appear and seeks to stamp out root causes of issues. (No doubt the success is also augmented by a lot of good luck.) Examples of the underlying security technology include security and privacy review processes, code analysis tools, abuse detection, and security by construction such as frameworks to avoid cross-sit--scripting. When assessing attacks from top global actors, we do not know what we do not know, but suppose for the sake of argument that this success was not an illusion. Can the same techniques help with defense systems? Yes, but with the cautionary note that this is not as easy as it may look from outside.

At the other extreme, the "small scale" examples I have in mind include NC3 components, high-ne--worth financial control endpoints, incident response workstations, and management of personal keys. Even more than in protecting commercial infrastructure, with these systems you don't get a second chance to clean up and try again. Furthermore, the scope of action is small enough to enable a non-standard computing approach. In NC3, I hear there are some very well-built solutions from decades ago that are now at risk of modernization. In the civilian world, the best currently-active security people I know are mostly working on the large-scale systems, but their techniques are not necessarily the ones best suited for extraordinarily high security requirements. Skeptics should be able to verify the security of NC3, not just generalize from seeming good experience to date. Luck is not a strategy. We will need better use of formal methods, which I will not have room to cover here, and we will need many orders of magnitude smaller systems, as measured by lines of code, dependency graphs, numbers of authors, or overall architectural complexity.

The US and the world are best off if all the nuclear weapon powers have verifiably secure NC3. Otherwise, their leaders have a strong incentive to delegate decisions out to the edge, and that increases the odds of accidental or rogue launches. The security needs to be verifiable because nuclear states are the ones who have demonstrated at great cost that they are unwilling to take anyone else's word on existential matters. Furthermore, to avoid creating proliferation incentives the non-nuclear-weapon states also need to be convinced of NC3 security. So I claim it is in the US national interest to work with our adversaries and our allies to help them build systems that even we cannot hack into.

Nature of the Threat at Large-Scale

Modernization of very old IT systems may seem like an obvious improvement, though I would argue this may be a misapprehension for NC3. We all like the speed, ubiquity, low cost, and apparent relative security offered by modern large-scale systems. It is therefore worth saying a bit more about the security challenges that have been tackled in large-scale systems. Gmail traditionally (say,

ten years ago) experienced the following problems:

- phishing.
- poor endpoint security.
- poor communication security.
- unwise sharing.

These problems were hardly unique to Gmail; I just cite them as the ones we observed at the time as primary ways our users were being hurt. Today, there are good solutions available to mitigate those threats:

- U2F Security Key[2].
- ChromeOS[3] and many browser security improvements.
- SSL everywhere[4].
- low cost personal machines to enable physical isolation and local control.

Aside from insufficient adoption of these available solutions, there are three major remaining problems from a user perspective:

1. bugs
2. court-supervised but very-intrusive discovery processes by law enforcement and civil suits
3. very advanced attackers or rogue insiders

An example of risk #1 would be the Google Docs sharing bug of March 2009[5]. That case was particularly interesting for the detailed communication to users, enabled by infrastructure logs, describing exactly how they were affected by the bug. As typical in these cases, problems found and fixed internally are important, but not a major part of the threat from NC3 attackers.

Risk #2 is growing[6] and sometimes has yielded remarkable releases of confidential emails[7]. Again, this is an important problem but not so much for NC3.

Risk #3 is a very relevant concern for NC3. Such attacks are rarely successful but could be catastrophic. A cutting-edge area of security operations is implementing the “four-eyes principle”[8] in a productive way, and well worthwhile for discussion and sharing of experience amongst the various organizations grappling seriously with it.

An individual user option for further protecting Gmail has always been PGP[9]. It was made substantially easier to use correctly inside Gmail by the e2email project[10], but by the time that came out the few people who cared strongly were already using Signal. Most people don't seem to feel their email is sufficiently valuable to go to any additional effort to protect it with encryption, beyond TLS built into their mail client. That may be a good thing, because had they tried to adopt secure messaging, they would have discovered the downsides of degraded storage and organization. Some of us concerned by the three risks above decided perhaps a better architecture than email is ephemeral messages (such as Signal) supplemented by a network file system with stronger security and less error-prone sharing. See upspin.io[11] for one solution.

Cloud providers do not demonstrate that their security is strong in absolute terms, only that it comprehensively meets baseline standards, that it includes a variety of defenses that outclass those of traditional in-house IT departments, and that there are few or no reported losses. So reasonable Cloud buyers trust, based on what little information they have, that the Cloud is secure relative to any alternatives that can support the tremendously diverse and interconnected applications needed for modern society.

The success of AWS, Azure, GCP and the widely used applications built upon such infrastructure have led some government agencies, financial institutions, and others try to emulate on the cheap. Don't underestimate the vast resources needed, and don't overestimate the security achieved.

In summary, I claim Gmail security is reasonably satisfactory, and certainly much improved over ten years prior. One need not despair that cybersecurity is hopeless. The security lessons have some, but rather limited, applicability to NC3.

A Vision of a Simpler Future

Even if such clouds are the best modern computing choice for most civilian and military systems, nevertheless I claim it is not the way to achieve verifiable security for NC3. We need to return to first principles and specifically to the rule "Complexity is enemy of security"[12]. The best statement of the rule I know is from Tony Hoare, inventor of Quicksort, who said in his 1980 Turing Award Lecture:

...there are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult. It demands skill, devotion, insight, ...and a willingness to accept ...a compromise when conflicting objectives cannot be met. No committee will ever do this until it is too late.

In short, I believe the right direction for critical modern NC3 components is a computer that more closely resembles a Unix machine of the late seventies than Cloud infrastructure or even a Linux desktop of today.

To give you a taste of what I mean, this essay was written over an ssh session using the vim editor and markdown formatting. Today's window systems (and to a lesser extent word processors) descend from an earlier era of optimism about our ability to write huge programs, and I don't have complete confidence in their success at retrofitting security. It may not be feasible to live productively for most of life without such systems, but for special high-value tasks we need to experiment with downsizing to systems of verifiable size. Another rule for "radically smaller" is to avoid running any code not explicitly installed. That implies no Javascript browser and no file formats with active content. Typical applications today have far too many dependencies and are written in languages too difficult for adequate security review.

We should not deceive ourselves that a radically smaller and more secure general-purpose computer will be broadly applicable. There will be extreme disadvantages, such as an inability to run most software (open source or not) and a requirement for operator training since some familiar UI elements will be removed. In the NC3 context, for example, I expect that modernization of sensor platforms and the data fusion tools that enables situational awareness are inherently complex, and can only be accomplished with complicated software.

In contrast, I argue that we should build very simple communication systems to be used by the President or other national leaders to talk amongst themselves and with their advisors in and out of

government or to issue commands to strategic military units. These hotlines should be modernized to take advantage of modern fabrication, but they need the extraordinary reliability and security that justifies effort toward radical simplicity.

There is plenty of remaining technical challenge in building a hotline in modern technology. Content confidentiality and integrity are easy enough, but harder is designing for endpoint discoverability and resisting traffic analysis and denial-of-service. Finding the right compromise to achieve simplicity will require Hoare's" skill, devotion, and insight."

Since there is no broadly applicable business case, due to the small market and significant effort, the most credible way to get the necessary engineering effort is to break off this piece from the larger NC3 or other motivating scenarios and allow part of our military engineering teams to participate in an open source, global, loosely coordinated effort. Open source's free rider problem is real but is mitigated here because if you don't participate, you're likely to misconfigure the system through lack of understanding. You can't pay someone to go to the gym as your substitute.

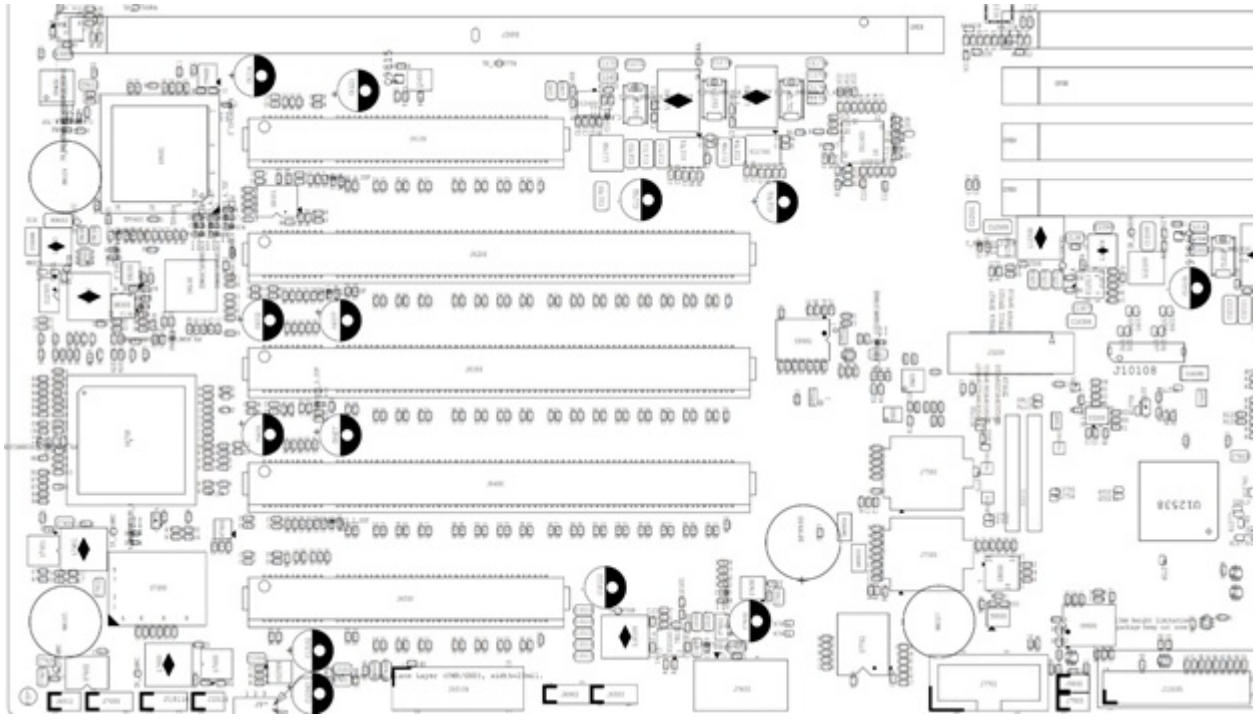
This open-source collaboration implies running security-critical software not written by US citizens. There are a lot of smart people in the world, they don't all live in this country, and not all the top-cleared citizens are trustworthy. So yes, let's build a security review system that is so strong that we are comfortable running software originally written by adversaries. There is an analogy with modern[13] approaches to network security. You can only convince your staff to treat all networks as compromised if you give them the tools and responsibility to live safely on the open Internet. They are going to anyway, so plan for it.

As a confidence building measure, it would be great to build and operate a shared public key server adopted by and jointly operated by competing militaries. Everything in such a system is open, verifiable, and easily backed up, and yet there is enough at risk to focus the mind. Building upon that, let us then jointly construct open-source hotlines, scrutinized by bitter adversaries who nevertheless agree to adopt them.

An Aside on Supply Chain Security for Small Systems

These new, simpler systems should be open source all the way down through firmware to the board schematics, such as with the TalosII[14] secure workstation hardware. Especially in a multi-country context, but even in a US-only context and for highly influential customers, binary blobs[15] make verification nearly impossible because there are never enough expert reviewers. But even if you do not accept the prediction of open hardware on the basis of Moore's Law ending[16] you should adopt it for verifiable security.

Recent media reports speculate on hardware insertions somewhere in the supply chain.[17] It is hard to assess how often this happens in the real world, but certainly we should all be concerned with whether we have any ability to detect such attacks someday when they do begin. When beginning to prepare this talk, I intended to apply modern AI image processing to compare the part placement schematic of my TalosII mainboard with a photograph.



Also, since we were asked to comment on AI: in a system designed to be simple enough to be security reviewed, the training algorithms of deep learning may be too complicated to include, but we may still be able to execute the relatively simple matrix multiplies for the derived models.

In looking at these images, it quickly became apparent that an attacker would just swap out the socketed BMC and Boot flash devices or use the write-enable switch to overwrite them. There is little practical point looking for fancier attacks yet.

Secure boot technology is understood; for example, the Chromebook this essay is being typed on has a good start. More broadly, we need a verifiable firmware base. In the u-root approach,[18] the kernel and the compiler toolchain are the only binaries in the boot system; everything else is compiled from understandable source at system initialization and yet start-up is fast. Not that even a compiler binary is without concern: a full solution here should use known defenses against the "trusting trust" attack.[19]

The chosen operating system kernel must be small enough, and changing slowly enough, for security review. Yet it needs to change rapidly enough to patch promptly any vulnerabilities and to install important new defense mechanisms as they become available. Some tiny version of Linux is the

obvious candidate, though past efforts were motivated by embedded hardware limitations rather than security and they may not represent the best pruning. Perhaps OpenBSD a good choice.

The vast collection of dependencies in userland applications is the most important opportunity for reduction of attack surface. This part of the software supply chain is our biggest risk, as the recent event-stream npm exploit illustrated.[20] Not even author-signed source and a verified compiler would protect against that exploit. The only hope is to radically minimize dependencies and establish a software review process that can watch all changes.

This whirlwind tour of current efforts to build a more verifiable small system illustrates the preliminary nature of our vision of a simpler future. I welcome discussion of such the proposed hotline approach.

Allow me to close with one last quote from Tony Hoare's talk, as he was reflecting on a time he was participating in a technical effort that he could see was taking on more than it should:

At first I hoped that such a technically unsound project would collapse but I soon realized it was doomed to success. ... There is nothing [I] can say that will stand against the flood of a hundred million dollars. But there is one quality that cannot be purchased in this way---and that is reliability. The price of reliability is the pursuit of the utmost simplicity. It is a price which the very rich find most hard to pay.

Acknowledgments

Most of all I thank the brilliant people at Bell Labs and Google who taught me security and wrangled systems complex beyond imagination. Ron Minnich convinced me that it is possible to avoid binary blobs.

I also thank Jim Gettys and others at Muinín for discussions on supply chain security. I like the company motto: Ná cuir muinín i gclaíomh briste. Irish proverb: Do not put your trust in a broken sword.

III. ENDNOTES

[1] Rachel Potvin and Josh Levenberg, "Why Google Stores Billions of Lines of Code in a Single Repository", Communications of the ACM 59:7, July 2018.

[2] http://En.wikipedia.org/wiki/Universal_2nd_Factor http://landing.google.com/advancedprotection/; Emma Dauterman et al., *True2F: Backdoor-resistant authentication tokens*, IEEE S&P (Oakland) 2019.

[3] http://Chromium.org/chromium-os.

[4] http://Letsencrypt.org/2014/11/18/announcing-lets-encrypt.html; browser UI research; engineering of servers.

[5] http://Cbc.ca/news/technology/google-docs-shares-users-private-files-by-accident-1.838830

[6] http://Transparencyreport.google.com/user-data/overview.

[7] http://Fortune.com/2018/11/26/facebook-uk-parliament-email-seizure/.

[8] http://En.wikipedia.org/wiki/Two-man_rule.

[9] Alma Whitten and J. D. Tygar, “Why Johnny Can’t Encrypt”, USENIX Security 2009.

[10] http://Github.com/e2email-org/e2email.

[11] http://Upspin.io

[12] Dan Geer [IEEE Security & Privacy, Nov/Dec 2008] was not the first to say it, but says it well and adds an illuminating analogy to the financial fiasco of the time, along with the caution "Nature is an existence proof that complexity isn’t the enemy of life".

[13] http://Beyondcorp.com

[14] http://Wiki.raptorcs.com/wiki/Category:Documentation.

[15] http://En.wikipedia.org/wiki/Binary_blob.

[16] Andrew “bunnie” Huang,The Hardware Hacker, No Starch Press, 2017.

[17]Trammell Hudson, “Modchips of the State”, 35C3 2018, http://youtube.com/watch?v=C7H3V7tkxeA.

[18] Ron Minnich et al., "Replace your exploit-ridden firmware with a Linux kernel" http://schd.ws/hosted_files/osseu17/84/Replace%20UEFI%20with%20Linux.pdf; See also http://threatpost.com/uefi-rootkit-sednit/140420/.

[19] David A. Wheeler, http://www.acsa-admin.org/countering-trusting-trust-through-diverse-double-compiling/

[20] http://Blog.npmjs.org/post/180565383195/details-about-the-event-stream-incident.

IV. NAUTILUS INVITES YOUR RESPONSE

The Nautilus Asia Peace and Security Network invites your responses to this report. Please send responses to: nautilus@nautilus.org. Responses will be considered for redistribution to the network only if they include the author’s name, affiliation, and explicit consent

---

View this online at: https://nautilus.org/napsnet/napsnet-special-reports/security-at-extreme-scales/

Nautilus Institute
608 San Miguel Ave., Berkeley, CA 94707-1535 | Phone: (510) 423-0372 | Email: nautilus@nautilus.org