



Q2 • 2022

ThinkstScapes Quarterly



THINKST
SCAPES

<https://thinkst.com/ts>

Brought to you by



**Most companies find out way too late
that they've been breached.**

Thinkst Canary changes this.

Canaries deploy in under 4 minutes
and require 0 ongoing admin overhead.

They remain silent until they need to chirp,
and then, you receive that single alert.

When.it.matters.

Find out why some of the smartest security teams
in the world swear by Thinkst Canary.

<https://canary.love>

Contents

Introduction	4
Themes covered in this issue	5
Security of networks – modern and legacy	6
I am become loadbalancer, owner of your network	7
Evil Never Sleeps: When Wireless Malware Stays On After Turning Off iPhones	8
AirTag of the Clones: Shenanigans with Liberated Item Finders	9
Are Blockchains Decentralised?	10
Languages and their ecosystems	11
What Log4j teaches us about the Software Supply Chain	12
Kani Rust Verifier	13
Cross-Language Attacks	14
Software Updates Strategies: A Quantitative Evaluation Against Advanced Persistent Threats	15
Deep dives into deep places	16
AMD Secure Processor for Confidential Computing Security Review	17
Living Off the Walled Garden: Abusing the Features of the Early Launch Antimalware Ecosystem	18
A Kernel Hacker Meets Fuchsia OS	19
Nifty sundries	20
Adaptive Multi-objective Optimization in Gray-box Fuzzing	21
Cooper Knows the Shortest Stave: Finding 134 Bugs in the Binding Code of Scripting Languages with Cooperative Mutation	22
Bypassing CSP with dangling iframes	23
Bypassing Dangling Markup Injection Mitigation Bypass in Chrome	23
Pre-hijacked accounts: An Empirical Study of Security Failures in User Account Creation on the Web	24
Conclusion	25

Introduction

Welcome to the Q2 2022 edition of ThinkstScapes! This issue focuses on content released, published, or presented since the publication of the Q1 2022 quarterly release.

It's worth noting that even with conferences rescheduled to this quarter from the last, a larger share of featured content was originally released in the form of blog posts. This may be due to researchers delaying their publication until one of the three Las Vegas conferences (Black Hat USA, DEF CON, or BSidesLV); regardless, high-quality blog posts and technical reports covered content in almost every selected theme.

As a reminder: We would appreciate your help in catching any interesting work that may have fallen through the cracks – any papers, presentations, or blog posts are welcome. Please send them to ts@thinkst.com!

This issue includes talks drawn from the following conferences (and 150 security blogs):

Conference/venue name	Number of publications reviewed
NDSS Symposium	83
Black Hat Asia	34
LangSec SPW	11
RSA Conference USA	389
CanSecWest	18
REcon	23
High Confidence Software and Systems Conference	33
Hardwear.io USA	16
TyphoonCon	16
IEEE Workshop on Offensive Technologies	11
Kernelcon	36
TROOPERS	34
BSides Knoxville	14
Total	718

Kloof Corner, Cape Town, South Africa. Photo by Tim Johnson on Unsplash.



As always, Thinkst Labs is happy to notify you when we release a new issue. Sign up on the [ThinkstScapes homepage](#) where you can also find a link to the audio summary of this issue.

Themes covered in this issue

SECURITY OF NETWORKS – MODERN AND LEGACY

Networks have traditionally been a field focused primarily on the optimisation and betterment of existing technologies. It is interesting to see multiple works covering network security from different perspectives; both supposedly-invisible portions of the legacy network stack, and completely novel networks that have emerged relatively recently. This theme explores attacking load balancers that terminate encryption, the network and devices that interact with Apple's new *Find My* feature, and a look at the Bitcoin network from a variety of perspectives.

LANGUAGES AND THEIR ECOSYSTEMS

New languages that are designed with security as a first-class feature are gaining popularity in products large and small. Along with the language itself, new languages are bundling build systems and dependency management into the core toolchain. This theme looks at novel work both in how those languages and ecosystems can improve security – and instances where they add attack surface.

DEEP DIVES INTO DEEP PLACES

Work in this theme is centred around low-level portions of a system, from a CPU component to kernel security. In addition to highlighting security issues, the process by which each of the researchers learned about and instrumented their respective targets is educational for others tackling novel environments. From an extensive review of AMD's security hardware, to Microsoft's anti-malware privilege level and a Linux kernel researcher exploring another OS, work in this theme provides more than just a list of fixed security vulnerabilities.

NIFTY SUNDRIES

As always, there are some papers that do not fit precisely into any emergent theme for the issue, but still warrant inclusion. This quarter includes work on applying multi-objective optimization on fuzzing, finding vulnerabilities in document parsers, some interesting interactions with parsing HTML, and finally a new type of account hijacking that is made possible by services adding SSO integrations.

Security of networks – modern and legacy

- ✓ *I am become loadbalancer, owner of your network*
- ✓ *Evil Never Sleeps: When Wireless Malware Stays On After Turning Off iPhones*
- ✓ *AirTag of the Clones: Shenanigans with Liberated Item Finders*
- ✓ *Are Blockchains Decentralised?*

The sun sets over Nature Valley in South Africa. Photo by redcharlie on Unsplash.

I am become loadbalancer, owner of your network

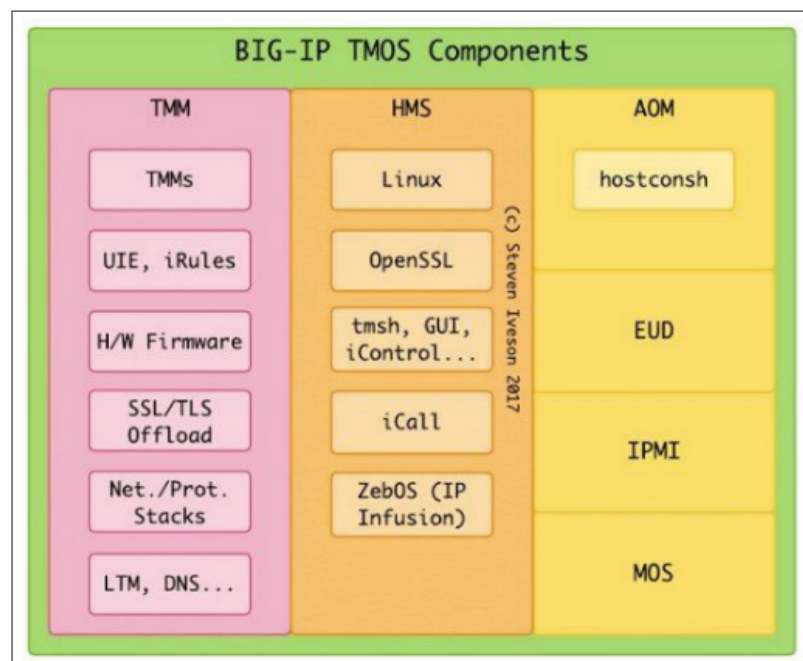
Author: Nate Warfield

[Slides](#)

This work looks at both a recent exploit for F5 BIG-IP load balancers (that allow an unauthenticated attacker remote code execution on the device) and a primer for how to avoid detection during post-exploitation. These load balancers usually are used to terminate SSL/TLS, so they can see or modify all traffic unencrypted. While the management is supposed to be entirely out-of-band, a number of serious vulnerabilities over the last few years has highlighted weaknesses on the user-facing traffic plane.

The most recent vulnerability presented (from May 2022) fits into a single tweet, but once code execution is acquired, next steps are more complicated due to F5's remote logging and synchronisation of configuration – if an attacker makes the wrong change, it will be replicated across the network and could cause a massive impact, alerting defenders quickly. Finally, the researcher highlights how F5's own knowledge base offers a procedure for gaining persistence by marking a script as a startup service.

Figure 1: A diagram of the software components on an F5 BIG-IP device.



TAKEAWAYS:

- ✓ **Load balancers are supposed to be unnoticeable in a network path**, yet this work shows they are a ripe target with significant impact. By design they must act as a parser for untrusted user input, and are positioned in a place of privilege in the stack. Unlike switches that have well-defined rules for routing IP traffic, load balancers must run custom logic and therefore expose more attack surface. Carefully consider every component in the path between an attacker and their target – more than just the endpoint may be in scope.

Evil Never Sleeps: When Wireless Malware Stays On After Turning Off iPhones

Authors: Jiska Classen,
Alexander Heinrich,
Robert Reith, and
Matthias Hollick

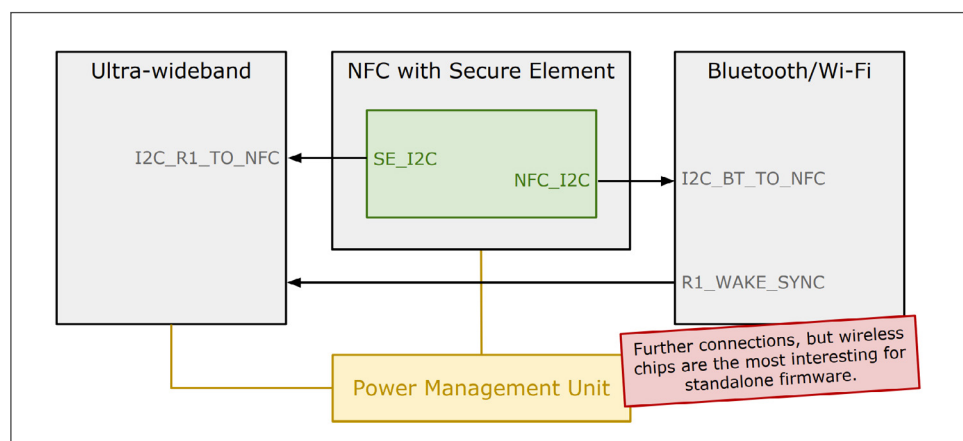
[Slides](#)

[Paper](#)

Building on past work exploring Bluetooth firmware and embedded wireless stacks featured in the [Q3 2021 ThinkstScapes edition](#), this paper explores modern iPhones' low-power mode (LPM). LPM is a mode of operation entered either when an iPhone is turned off by the user, or when the battery is insufficient to keep iOS running. To support the **Find My** Location feature, as well as some payment or access cards/keys, LPM keeps the ultra-wideband, NFC and Bluetooth radios active.

While the firmware for both the NFC and UWB radios are signed, the Bluetooth module allows the device to patch its firmware with only CRC integrity checks. On a jailbroken iPhone, the host control interface allows for the main application processor to send updated code values to the radio. Exploring the functionality provided in LPM and how it is implemented via the wireless interfaces provides some possible attacks on changing the **Find My** behaviour, e.g., to allow an attacker to track a device turned off by its owner. Additionally, a few limitations are highlighted based on the pre-cached **Find My** advertisements that cannot be regenerated until after iOS has booted and the user authenticated.

Figure 2: A high-level schematic of the components on modern iPhones that remain on for multiple hours after a user- or low power-prompted shutdown of the device.



TAKEAWAYS:

- ✓ **While this specific attack will have little real world impact**, the work highlights the complexity of modern devices, and how our notions of on and off must evolve alongside the technology's interpretation. A software update can drastically change the behaviour of the device. Most of the security and privacy concerns in this work were specific to opaque messaging, but seeing how devices can change with an OTA update shows the flexibility of these new platforms.

AirTag of the Clones: Shenanigans with Liberated Item Finders

Authors: Thomas Roth,
Fabian Freyer, Matthias
Hollick, and Jiska Classen

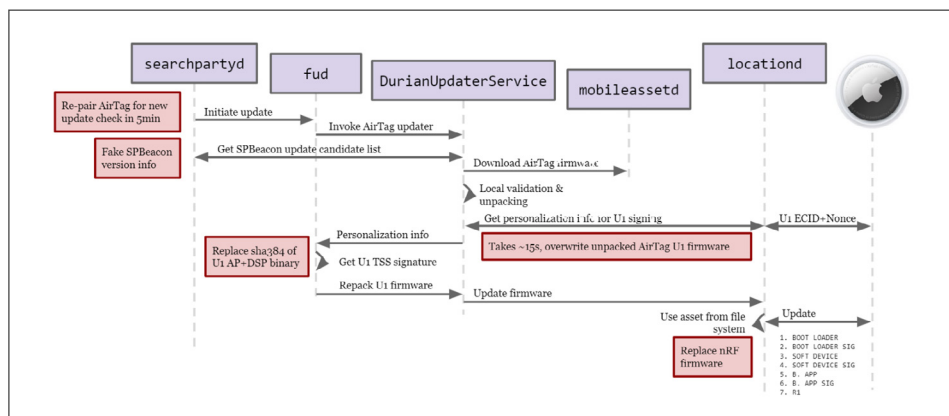
[Paper](#)

[Code](#)

Apple AirTags are low-cost trackers that make use of a powerful ad hoc network (the *Find My* network) to locate the tags around the world, even in areas out of range of conventional wireless networks. As most Apple hardware devices are automatically enlisted to act as a part of the network, and Apple devices are commonplace, the *Find My* network may be one of the largest wireless networks outside of traditional cellular or WiFi.

While there has been past research on how AirTags may be used for stalking or other malicious purposes, this work explored the hardware itself, reverse engineering and building open-source tooling to assist in exploration of how the devices operate and interact with other nodes in the *Find My* network. Beginning with a low-cost power glitching attack, the researchers were able to gain access to the firmware of both the device and the device's new ultra-wideband radio that allows for fine-grained location reporting. Then the researchers were able to change some of the behaviour of the device including serial number, audio alerts, etc. Additionally they demonstrated the ability (see Figure) to downgrade the firmware, or desynchronize the firmware versions between the device and the UWB radio controller.

Figure 3: A high-level transition diagram of how the Apple Airtags process firmware updates—this research shows that downgrades or mismatched versions are possible.



TAKEAWAYS:

- ✓ While the impact to end-users of this work is limited, the tools provided allow others to explore this new network. Outside of fixed cellular and WiFi installations, few wireless networks have offered the coverage and functionality of the *Find My* network. Exploring how these ad hoc networks interact with other devices and society will offer interesting outcomes.

Are Blockchains Decentralised?

Authors: Evan Sultanik, Alexander Remie, Felipe Manzano, Trent Brunson, Sam Moelius, Eric Kilmer, Mike Myers, Talley Amir, and Sonya Schriener

[Blog](#)

[Paper](#)

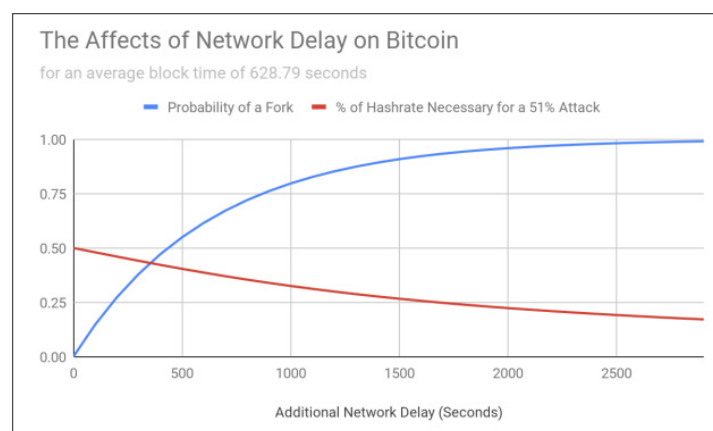
[Audio](#)

There has been extensive research into the cryptographic promises made by the technology underlying cryptocurrencies; this work looks at the oft-repeated claims of decentralisation. Breaking down (de)centralisation into multiple aspects: authoritative, consensus, motivational, topological, network and software centrality, each is examined to understand the risks to the blockchain by a small number of malicious entities.

Excluding the cryptographic protocols, there were a number of issues that could be (or have been) used to subvert the network, from an entity gaining control of Tor exit nodes (which route a majority of Bitcoin traffic) to modify or drop blockchain traffic, to malicious transactions exploiting differences in versions of the software across the network to create forks. While some issues can be addressed, such as the version disparity of nodes, others require new research to eliminate (e.g., Sybil cost to ensure decentralisation).

Many of the issues highlighted are accompanied by a real-world example of these attacks occurring, or statistical deviations observed in the blockchain that indicates non-standard or private interconnections between supposedly independent nodes. The results indicate that a majority of individuals running nodes with the aim of strengthening the security of the network contribute nothing to its overall security, and private or undocumented functionality plays a key role in the networks veracity.

Figure 4: A chart ([sic] the title) showing how introducing network delays in the Bitcoin network can increase chances of a fork, and thereby reduce computational power needed to claim a majority in the network.



TAKEAWAYS:

- ✓ **Breaking down the technical, social, and incentive-based models** of any system allows for better analysis of the claims of centralisation. This work reminds us that despite the (almost) accurate claims of needing 51% of the computational power to subvert the Bitcoin network, you only need to subvert one of the four developers' accounts, or two of the four major mining pools.
- ✓ **Ken Thompson's "Reflections on Trusting Trust"** is especially relevant to the cryptocurrency space: cryptocurrencies require trusting a large number of individuals who may not have the same incentives to act in a manner that benefits the end user. This paper highlights that the group is more diverse than initially thought, including Tor nodes, developers of shared dependencies and closed-source mining frameworks.

Languages and their ecosystems

- ✓ *What Log4j teaches us about the Software Supply Chain*
- ✓ *Kani Rust Verifier*
- ✓ *Cross-Language Attacks*
- ✓ *Software Updates Strategies: A Quantitative Evaluation Against Advanced Persistent Threats*

What Log4j teaches us about the Software Supply Chain

Author: Stephen Magill

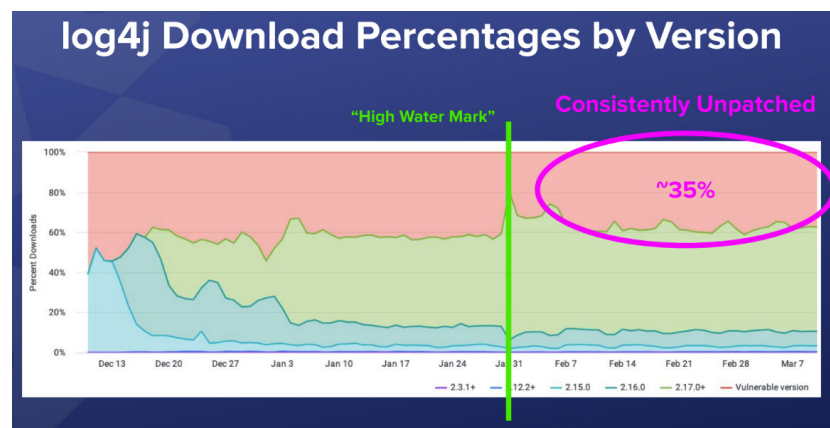
[Slides](#)

[Video](#)

This keynote explored the response to the Log4Shell exploit released late in 2021 from the perspective of the Maven Central repository's owners. The exploit targeted a vulnerability in Log4j, a popular dependency in many Java-based applications (while ~70k applications directly depend on Log4j, ~175k transitively depend on it – in one application, there are 30 levels of dependencies separating the application from Log4j).

The Log4j incident was complicated by an incomplete initial patch; although the projects that were early to update were quick to adapt to changing versions, the long tail of projects that never updated persisted indefinitely. Finally the author presented a cautionary tale of modern software supply-chain attacks where versions of dependencies were either corrupted or typo-squatted. Automated build and dependency systems still leave some thorny challenges left to solve.

Figure 5: A chart showing downloads from Maven central of vulnerable and patched versions of Log4j after the release of the Log4Shell exploit.



TAKEAWAYS:

- ✓ **The data presents the best view possible** of the response to the Log4j vulnerability – projects that used automatic build systems to manage dependencies. Even with this skewed data, for a vulnerability that was in the mainstream news, a significant percentage of downloads continue to be for vulnerable versions. For less amplified weaknesses, the percentage of updated projects is much lower.
- ✓ **A software bill of materials (SBOM)** only provides the most basic information that could indicate a vulnerability's applicability. SBOMs can over-approximate applicability by conflating inclusion with use, or use in a manner that allows exploitation, resulting in overwhelming responders to a new bug. Unless the context of use is exposed, it is impossible to know if the 35% of projects still consuming vulnerable versions are in fact vulnerable, or perhaps are not logging any user-provided data.
- ✓ **Automatically tracking the most recent versions** can offer easier protection against zero-days, but must be coupled with robust testing and validation that the new version still works as expected, and does not come with any other "bonus" functionalities such as a DoS or credential exfiltration.

Kani Rust Verifier

Authors: Daniel Schwartz-Narbonne and Ziad Hassan

[Slides](#)

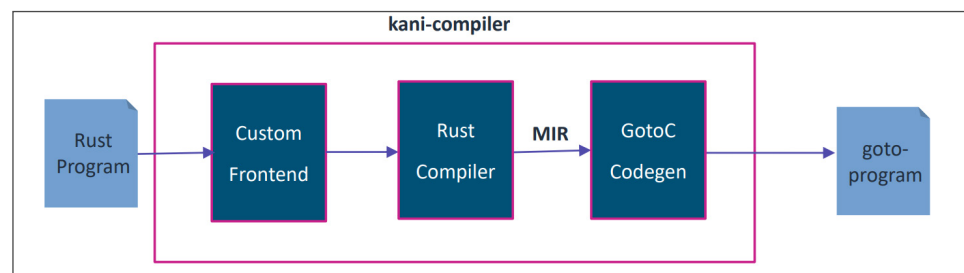
[Video](#)

[Code](#)

This work explores formal verification of a number of security-relevant properties in unsafe Rust blocks. In traditional Rust development, there is a large drop-off in safety for all code developed inside of an unsafe block that can propagate back into the safe/type-checked code. Kani works to even out that drop-off by using formal verification to guarantee the absence of certain weaknesses in unsafe code. By converting the Rust source into an intermediate representation ingestible by the CBMC model checker, multiple classes of bugs and additional semantic checks can be verified.

The open-source checker has been integrated into the developer workflow within AWS and follows a proven methodology to gain adoption: allowing developers to write specifications and checks in the language they develop in, embed those specifications into the code, and run the proofs as part of the CI/CD pipeline. Future development will add to the five classes of errors automatically checked and two types of developer-specified properties, providing closer parity to safe Rust when unsafe code is used in a predictable manner.

Figure 6: A diagram of the core component of Kani, the compiler from Rust to CBMC's goto-program intermediate representation.



TAKEAWAYS:

- ✓ **Many unsuccessful attempts have been made over the years** to harness the benefits of low-level code while offering (more) safety from traditional corruption attacks. Rust appears to have crossed that chasm, with adoption into the Linux Kernel and demonstrated everyday usage by tech powerhouses. It's clearly no longer a "toy-language" and should be worth considering for upcoming (suitable) projects.
- ✓ **While there are known security properties** associated with the type-checked "safe" Rust, unsafe Rust offers no such protections and even a small amount of unsafe code can introduce vulnerabilities into a largely safe codebase. Tools like Kani offer the ability to retain the safety of checked Rust with the flexibility offered by unsafe Rust – even with the listed conditions that Kani is unable to verify.
- ✓ **The workflow with which Kani is designed to integrate** should help increase its utilisation by targeting developers instead of a second verification team that inspects code after-the-fact. That AWS is able to get formal proofs for a portion of their C/C++ codebases is a testament to the ease of use and integration into the development lifecycle.

Cross-Language Attacks

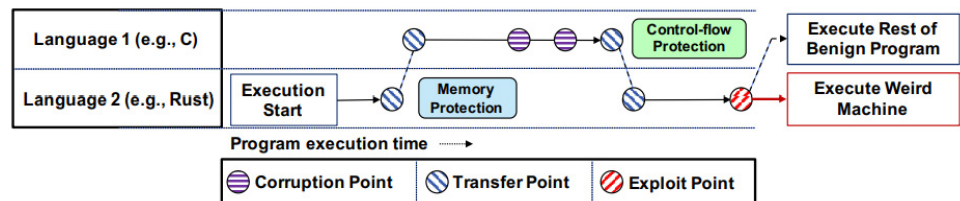
Authors: Samuel Mergendahl, Nathan Burow, and Hamed Okhravi

Paper

Rust and Go are modern languages that can offer significant security improvements at compile and run-time over legacy C/C++ codebases. A common strategy is for legacy codebases to be slowly replaced by software written in a newer and safer language, especially functionality that handles untrusted user input. If the input is sanitised in an environment with language and run-time checks on safety, the overall codebase's attack surface can be reduced.

This work looks at the security properties promised by both run-time protections on legacy code (e.g., ASLR, CFI, and DEP) and those built into more modern languages (focusing primarily on Rust). As seen in the figure, due to differences in those properties, a multi-language code-base can be more vulnerable to memory corruption attacks than a single-language application with best-practice mitigations in place. The majority of the attacks detailed stem from the ability of C/C++ code to change Rust's memory state: Rust code can then be circumvented to run arbitrary code as Rust does not have CFI. The CFI-protected C/C++ code is used to mutate the state of the Rust language environment, but the actual exploit is triggered within Rust as there are fewer run-time protections due to the more stringent compile-time guarantees – guarantees broken in a multi-language environment.

Figure 7: An overview of exploiting a hybrid C and Rust program.



TAKEAWAYS:

- ✓ **The common consensus** is that gradually porting a codebase to, or adding new components in a safer language is guaranteed to reduce attack surface. This work highlights that there are disparities in the security properties between the improvements made to protect legacy code and the built-in safety guarantees in safer languages that can open new vulnerabilities.
- ✓ **While this work does in fact highlight** that some safety guarantees offered by some programming languages can be violated, the overall burden on an attacker is likely increased over continuing to develop in legacy languages.

Software Updates Strategies: A Quantitative Evaluation Against Advanced Persistent Threats

Authors: Giorgio Di Tizio,
Michele Armellini, and
Fabio Massacci

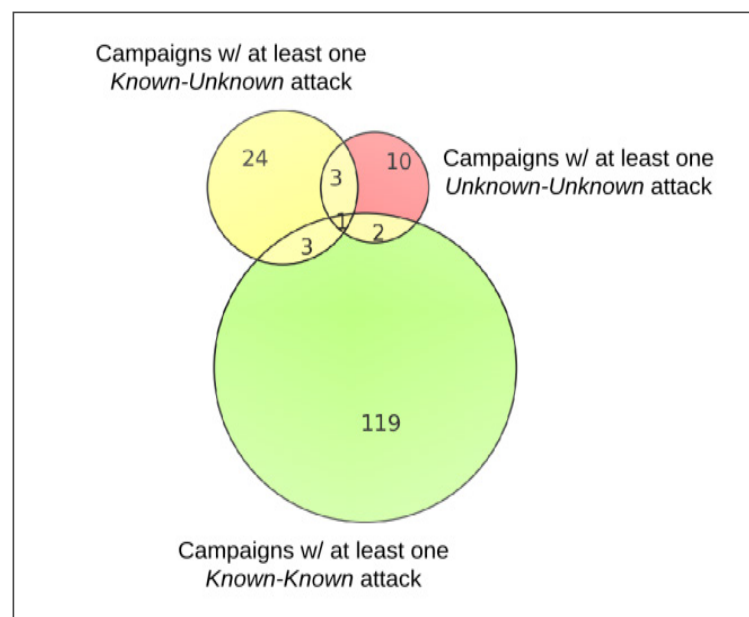
Paper

Data

This paper takes a detailed look at a dataset of APT attacks from 2008 to 2020. The authors build a database of APT campaigns and then present a methodology to analyse the attack vectors, vulnerabilities, and software exploited by 86 different APTs in more than 350 campaigns over 12 years. The database is publicly available on Github. The prioritisation of software updates proves to be a key factor to reduce the impact of these intrusions.

This is the first look at a public dataset to correlate APT campaigns, techniques used, CVEs, and vulnerable products. The authors observe that preventative measures, like rapid deployment of software updates (and their dependencies) can substantially reduce the probability of being compromised by an APT using public exploits for known vulnerabilities. The diagram below from the paper details the overlap of known and unknown vulnerabilities used in attacks. There is a clear trend towards known, exploitable vectors – thus the authors present a conclusion that improved patch management will help organisations, even against APTs.

Figure 8: A classification of APT Campaigns, the majority of campaigns exploited at least one vulnerability in a known-known attack (after publication by NVD and after reservation by MITRE).



TAKEAWAYS:

- ✓ **The long-standing approach of delaying patches** may need to be challenged in the face of data showing APT campaigns are leveraging these vulnerabilities to gain access to organisations. Security and IT teams may need to switch to faster patching models to keep in front of these known attacks. Additionally, we hope to see this set extended and built upon, as well as seek more contributions of public data on attacks to inform our software patching decisions.

Deep dives into deep places

- ✓ *AMD Secure Processor for Confidential Computing Security Review*
- ✓ *Living Off the Walled Garden: Abusing the Features of the Early Launch Antimalware Ecosystem*
- ✓ *A Kernel Hacker Meets Fuchsia OS*



Dolphins playing in the surf in Knysna, South Africa. Photo by redcharlie on Unsplash.

AMD Secure Processor for Confidential Computing Security Review

Authors: Cfir Cohen, James Forshaw, Jann Horn, and Mark Brand

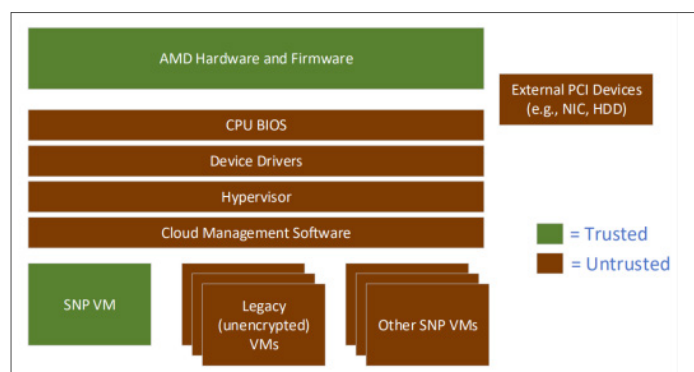
[Blog](#)

[Paper](#)

Following last quarter's theme on Confidential Computing, this work by Google and AMD provides a deep security review of AMD's security components (both hardware and firmware). The goal of AMD's SEV-SNP technology is to allow end-users to create virtual machines that the host (i.e., cloud provider) has no access to – the end user only trusts the AMD CPU and associated firmware. This work takes a deep look at the security of the trusted components and finds numerous vulnerabilities (which have been fixed). While the researchers had access to the source code underlying this system, they explain in their report all the ways they were able to review the security without immediately using source access.

Many of the discovered vulnerabilities only impact the AMD platform, but the methodologies for their discovery are generic enough to apply to other black-box systems and cryptography code. Highlighting how to use Wycheproof and PCIe Screamer to interact with the cryptographic functionality and filtering on the PCI bus, respectively, the experimental setup is well-documented for others to apply the same techniques to other platforms. The researchers detail their approach for exploring this system, and despite the issues discovered, remark an overall high security posture.

Figure 9: A classification of APT Campaigns, the majority of campaigns exploited at least one vulnerability in a known-known attack (after publication by NVD and after reservation by MITRE).



TAKEAWAYS:

- ✓ **While few entities are able to get the level of access** to a CPU vendor's source code, this report shines light on both the white-box and black-box approaches of performing an in-depth security analysis of a CPU component. This report will help researchers explore other platforms even without the privileged access afforded to Google.
- ✓ **This report highlights the complexity** in both modern processors as well as the confidential computing primitives that are the trust anchors for end users. Even these well-tested and competently developed systems can have issues. Opening their system to trusted third party review should help build trust in platforms built upon their confidential computing primitives.
- ✓ **It is generally accepted** that "rolling your own crypto" is a recipe for disaster; this report shows that even well-designed implementations of accepted cryptographic protocols can have subtle weaknesses that could result in total failure. These testing methodologies can be used to vet other implementations that are deployed widely to verify their adherence to properly handling edge-cases.

Living Off the Walled Garden: Abusing the Features of the Early Launch Antimalware Ecosystem

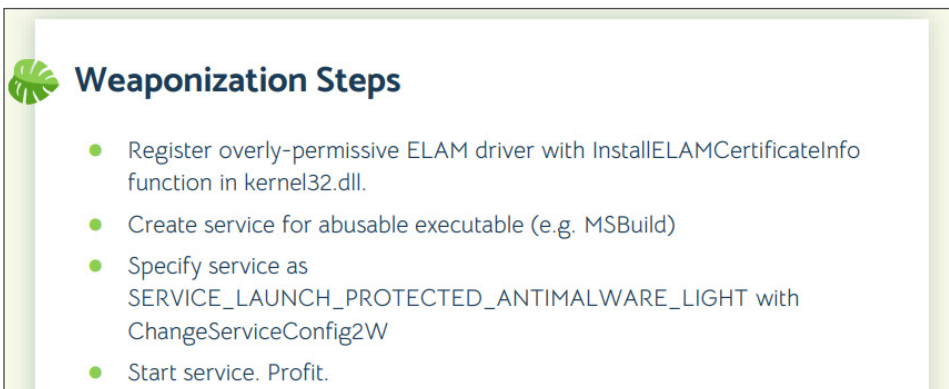
Author: Matt Graebar

Slides

This work explores the protected execution environment in Microsoft's Windows that is used by anti-malware processes to prevent manipulation even by administrators. Processes protected in this manner cannot be stopped or debugged – in other words, this is a perfect runtime environment for a malicious process to persist. The researcher explored the protections in place (very restrictive organisational agreements with Microsoft) resulting in a certificate chain to sign processes that run in the protected mode.

By searching for drivers or processes that were signed by certificates in this chain on VirusTotal, the researcher was able to find a combination of an overly-permissive driver that could be subverted, and a Microsoft-signed application (MSBuild) that could be started in this protected mode. In order to persist arbitrary code, the signed application cannot be used to spawn a separate process, so the code is added as a MSBuild test property, allowing arbitrary code to be run in MSBuild's process space, while being protected by the OS. Once executed in this protected space, the arbitrary code was then able to evict or stop other processes in this environment – e.g., terminating the Windows Defender runtime.

Figure 10: A slide outlining the procedure to abuse a weak ELAM driver certificate chain to get code execution in Microsoft's protected process space.



Weaponization Steps

- Register overly-permissive ELAM driver with InstallELAMCertificateInfo function in kernel32.dll.
- Create service for abusable executable (e.g. MSBuild)
- Specify service as SERVICE_LAUNCH_PROTECTED_ANTIMALWARE_LIGHT with ChangeServiceConfig2W
- Start service. Profit.

TAKEAWAYS:

- ✓ **The castle-style model** of the hardened and highly protected execution environments can have serious ramifications when compromised. Like past examples (e.g., kernel, hypervisor, or SMM root-kits, etc.) the Early Launch Antimalware Ecosystem offers a compelling target to attackers who want to deeply control a system and prevent other defensive processes from snooping on them. The tug-of-war between locked down positions of privilege and open computing systems is shown here clearly – allowing end users to change the trusted certificates would go a long way towards preventing abuse.
- ✓ **VirusTotal has shown itself to be a source of information** for all types of users, from defenders querying a file discovered in their environment, to malware authors testing their evasion, to attackers looking for signed binaries to subvert. Much like Shodan, VirusTotal has become a staple for users of all types and familiarity with its capabilities can offer multiple rewards.

A Kernel Hacker Meets Fuchsia OS

Authors: Alexander Popov

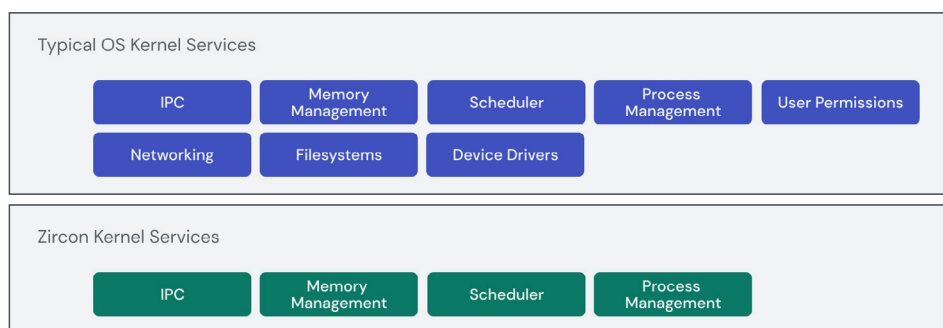
[Blog](#)

[Video](#)

In this blog post, the author, a Linux kernel contributor and security researcher, explores the microkernel-based Fuchsia operating system developed by Google. Despite the microkernel and capabilities architecture, vulnerabilities were discovered that allowed for code execution in kernel space. Due to the limited functionality provided by the kernel the post-exploitation steps had to explore new persistence options.

The author lays out their approach to building, debugging, and fuzzing the OS, which loads components via URIs that are automatically updated from a repository. In the limited time dedicated to this work, the author was unable to get a kernel fuzzer to operate, instead looking at how to exploit a UAF in the kernel. Along the way, they discover issues with KASLR and some of the capability checking logic – small, fixed issues, but reminders that a microkernel does not inherently mean guaranteed security.

Figure 11: A diagram showing the restricted subset of functionality in kernel space compared to a more monolithic operating system.



TAKEAWAYS:

- ✓ **While the specific weaknesses identified in this post** have little real-world impact due to the limited deployment of Fuchsia, the approach and discovered similarities between traditional OS kernels provide insights into the security of microkernels. The post-exploitation steps are especially insightful due to the reduction in kernel functionality and introduce readers to novel persistence techniques.



Despite the microkernel and capabilities architecture, vulnerabilities were discovered that allowed for code execution in kernel space.

Nifty sundries

- ✓ *Adaptive Multi-objective Optimization in Gray-box Fuzzing*
- ✓ *Cooper Knows the Shortest Stave: Finding 134 Bugs in the Binding Code of Scripting Languages with Cooperative Mutation*
- ✓ *Bypassing CSP with dangling iframes*
- ✓ *Bypassing Dangling Markup Injection Mitigation Bypass in Chrome*
- ✓ *Pre-hijacked accounts: An Empirical Study of Security Failures in User Account Creation on the Web*

A lion in Kruger National Park, South Africa. Photo by Vincent van Zalinge on Unsplash.

Adaptive Multi-objective Optimization in Gray-box Fuzzing

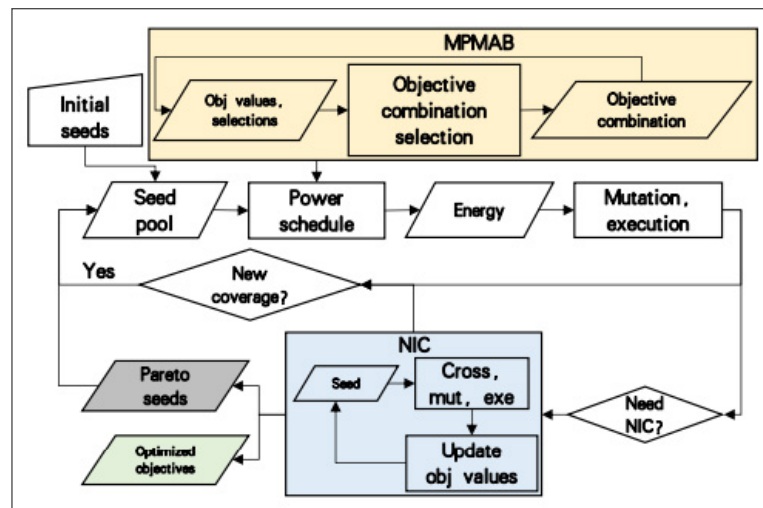
Authors: Gen Zhang,
Pengfei Wang, Tai Yue,
Xiangdong Kong, Shan
Huang, Xu Zhou, and Kai Lu

Paper

This work looked at the incentives driving mutation in fuzzers, and explored applying the mathematical principles of multi-objective optimisation to fuzzing. One of the most popular models in current use is coverage-guided fuzzing, popularised by the AFL toolsuite. The researchers explored different, sometimes competing objectives in fuzzing (e.g., power consumption, testcase runtime, coverage, etc.) and modelled it as a variant of the multi-armed bandit problem.

By bringing together the deep theory from optimisation of this style of game (exploring different options with unknown payoffs and costs) with vulnerability research tools, the authors' fuzzer was able to fairly consistently outperform existing state-of-the-art fuzzers both in terms of power consumption and bugs discovered. Using a benchmark of real programs with real bugs left in, their tool found more bugs than the competition, which included AFL++, SYMCC, and honggfuzz.

Figure 12: A diagram showing the process for test-case generation and mutation.



TAKEAWAYS:

- ✓ **As vulnerability research techniques become more mainstream** and treated as a suitable topic for academic study, there will be more rigorous application of computer science theory to optimise “hacker” tools. As this link between industry and academia strengthens, expect to see improvements from this collaboration.

Cooper Knows the Shortest Stave: Finding 134 Bugs in the Binding Code of Scripting Languages with Cooperative Mutation

Authors: Xu Peng, Yanhao Wang, Hong Hu, and Purui Su

[Slides](#)

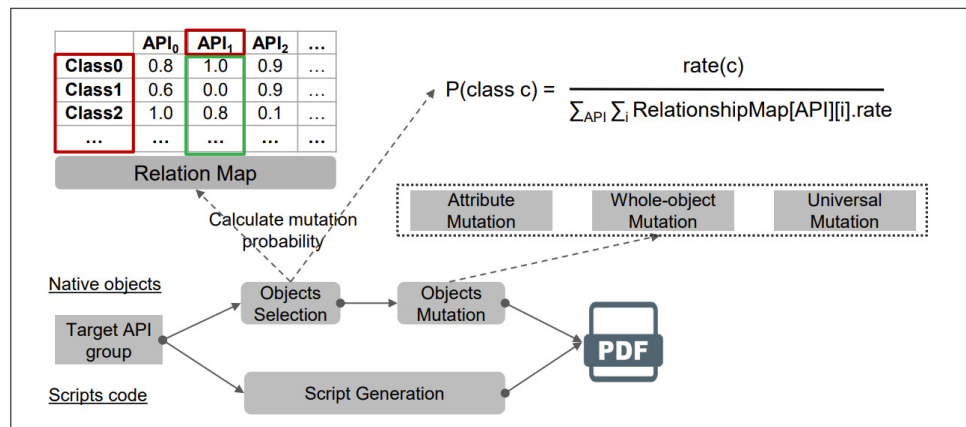
[Paper](#)

[Code](#)

This work explored the scripting execution engines embedded into various document formats (specifically PDF and Microsoft Word for this work) to add dynamism to rendered content. Rather than exploring the scripting engines in isolation, the researchers used an input corpus to try to identify correlations between data objects in the document format and the scripting functionality that interacted with those objects. The result was an impressive 134 new bugs across real-world programs (Adobe Reader, Foxit Reader, and Microsoft Word) that may never have been discovered by single-sided exploration.

By combining mutation to objects and related scripted operations, the tool was able to surface bugs where the conditions in either the data structure or code was insufficient to trigger, but in concert were easily detected. As an example, an object with no properties being accessed by the scripting environment creates a use-after-free in the JavaScript engine. The tooling has been released as open-source, and is extensible to other programming languages and related document formats.

Figure 13: A high-level diagram of the developed mutation tool Cooper, which looks for relationships between data and scripting environments that operate on that data to expose bugs through mutual mutation.



TAKEAWAYS:

- ✓ **It should go without saying** that adding a scripting execution environment to a complex data format parser will result in security concerns. PDF is already incredibly complicated to parse when well-formed, and numerous bugs have surfaced when handling malformed data – adding in the interactions between scripting engines and data will continue to expose more for quite some time.
- ✓ **The concept of relationship-guided mutation is powerful** as seen here, but broadly applicable to many environments where data and the code operating on that data are both attacker-controlled. Watch for this technique being applied elsewhere with similarly impressive results.

Bypassing CSP with dangling iframes

Author: Gareth Heyes

Bypassing Dangling Markup Injection Mitigation Bypass in Chrome

Author: SeungJu Oh

```
<script>
function cspBypass(win) {
  win[0].location = 'about:blank';
  setTimeout(()=>alert(win[0].name), 500);
}
</script>
<iframe src="//subdomain1.portswigger-labs.net/bypassing-csp-with-dangling-
iframes/target.php?email=%22><iframe name=%27"
onload="cspBypass(this.contentWindow)"></iframe>
```

Figure 14: Example of a dangling iframe tag to inject a script and bypass the CSP.

Chrome supports built-in dangling markup mitigation in order to block requests containing restricted characters. Dangling markup mitigations built into the browser can help protect an application that does not properly filter or escape the > or " characters. For example, attackers can use crafted syntax to break out of quoted attribute values and the enclosing tag, and inject content they control.

The first article describes a Content Security Policy (CSP) mitigation bypass in Chrome. The author noticed one of their labs breaking with a specific version of Chrome. Using their tool, the Hackability Inspector, and probing for various iframe properties and injection opportunities, the author discovered that setting the iframe location property to "about:blank" allowed the bypass of the Chrome Mitigations to read and inject scripts into the cross domain iframe. CSP treats about:blank URLs as the same origin – however, when an attacker sets a cross domain iframe to about:blank, it becomes readable and writable by an attacker and is definitely not the same origin.

In the second article, the author found a situation where a dangling tag interfered with Chromium's protocol upgrading. Chromium has a security

feature that automatically attempts to upgrade unsafe HTTP protocols to HTTPS. For example, if the src attribute of an tag uses an HTTP scheme, the browser automatically upgrades the scheme to HTTPS. On a page served via HTTPS, a dangling img tag referencing a non-HTTPS source could lead to content from the page being leaked to an attacker.

TAKEAWAYS:

- ✓ **These types of attacks** may affect payment and transaction applications that depend on iframes being protected from being read and written to. The combination of an application vulnerable to injection, as well as the mitigation bypass, could lead to compromised applications.
- ✓ **These types of mitigation bypasses** have found their way past the product testing and mitigation testing. Despite being a known class of attack, these two articles show it's very much an active area of exploration when coupled with another arcane feature (about:blank or protocol upgrades respectively).

Pre-hijacked accounts: An Empirical Study of Security Failures in User Account Creation on the Web

Authors: Avinash
Sudhodanan and
Andrew Paverd

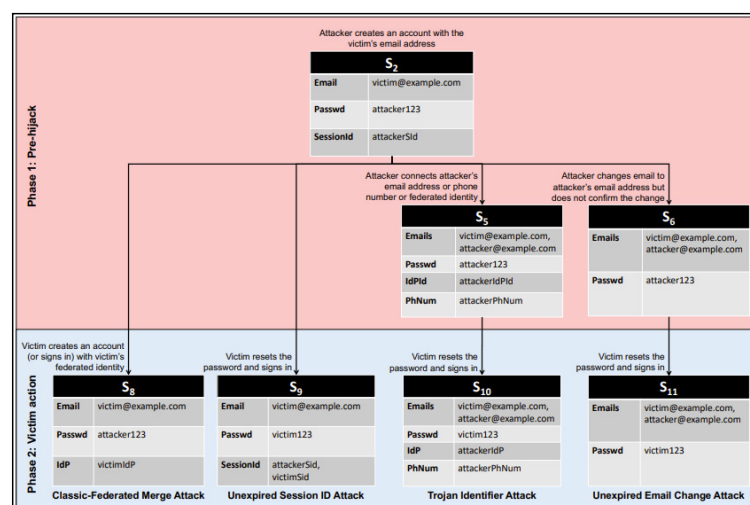
[Blog](#)

[Paper](#)

This work explored methods of hijacking accounts for online services via a novel approach of registering a victim's account prior to the victim, then waiting for the victim to use the account. For many web services that have both a legacy username and password authentication scheme, and have retrofitted their service to support SSO providers, there are ways where an attacker can retain access to an account without the victim knowing. The researchers built out a number of attack primitives, and manually explored half of the Alexa top 150 most trafficked websites (those excluded were region-specific variants of others included, sites without logins, or sites in other languages) to determine vulnerability to their identified primitives, of which a majority were impacted.

Some of the attack types relied on undefined behaviour when an account manually created with an email and password was merged with an SSO federated account with the same email. In some implementations the attacker could continue to use their manual credentials while the victim would use the SSO access. Other edge cases the researchers explored were with some services that allowed enterprise identity verification providers that did not verify the email addresses (e.g., creating your own provider in test account mode). This would then allow attackers to be treated as authoritative account owners despite having no access to the victim's email address.

Figure 15: A figure summarising some of the attack primitives discovered to pre-hijack a victim account.



TAKEAWAYS:

- ✓ **While the reputational risks** of not claiming an account have been explored and in some cases protected against, this work shows a new security risk of not claiming an account before an attacker.
- ✓ **SSO has spread across a diverse group of online services**, and this work shows that the details of how each service handles their retrofitting to support SSO matters immensely. Due to the simplicity of SSO, both during account creation and subsequent logins, it is easy to imagine pre-hijacked accounts netting numerous users who never set a password to their account.
- ✓ **The non-verifying IdP attacks** highlight the challenges of supporting the diversity of the enterprise identity management products while also relying on certain assumptions to offer SSO to the broadest customer base possible.

Conclusion

Despite research that is surely being saved for next quarter's "Hacker Summer Camp", the cadence of interesting work has continued apace, bolstered by a strong complement of blogs. It remains to be seen if this is a trend in the industry stemming from a long COVID-induced hiatus, or selection bias as the ThinkstScapes pipeline for collecting non-conference materials improves.

THREE THEMES EMERGED AT THE FOREFRONT OF INFORMATION SECURITY RESEARCH THIS QUARTER:

1. New looks at network security for legacy and novel networks
2. How modern programming languages and their environments shape software security
3. Framing how low-level researchers approach a new and foreign system target

As next quarter's issue will include one of the largest multi-events of the year, expect some gems held back for a dramatic reveal. While there are always a few talks that captivate the media's attention in the lead up to Hacker Summer Camp, check next quarter's issue for the field-shaping work that readers may not have caught.

