Empirical Security & Privacy, for Humans

UPenn CIS 7000-010

Introduction

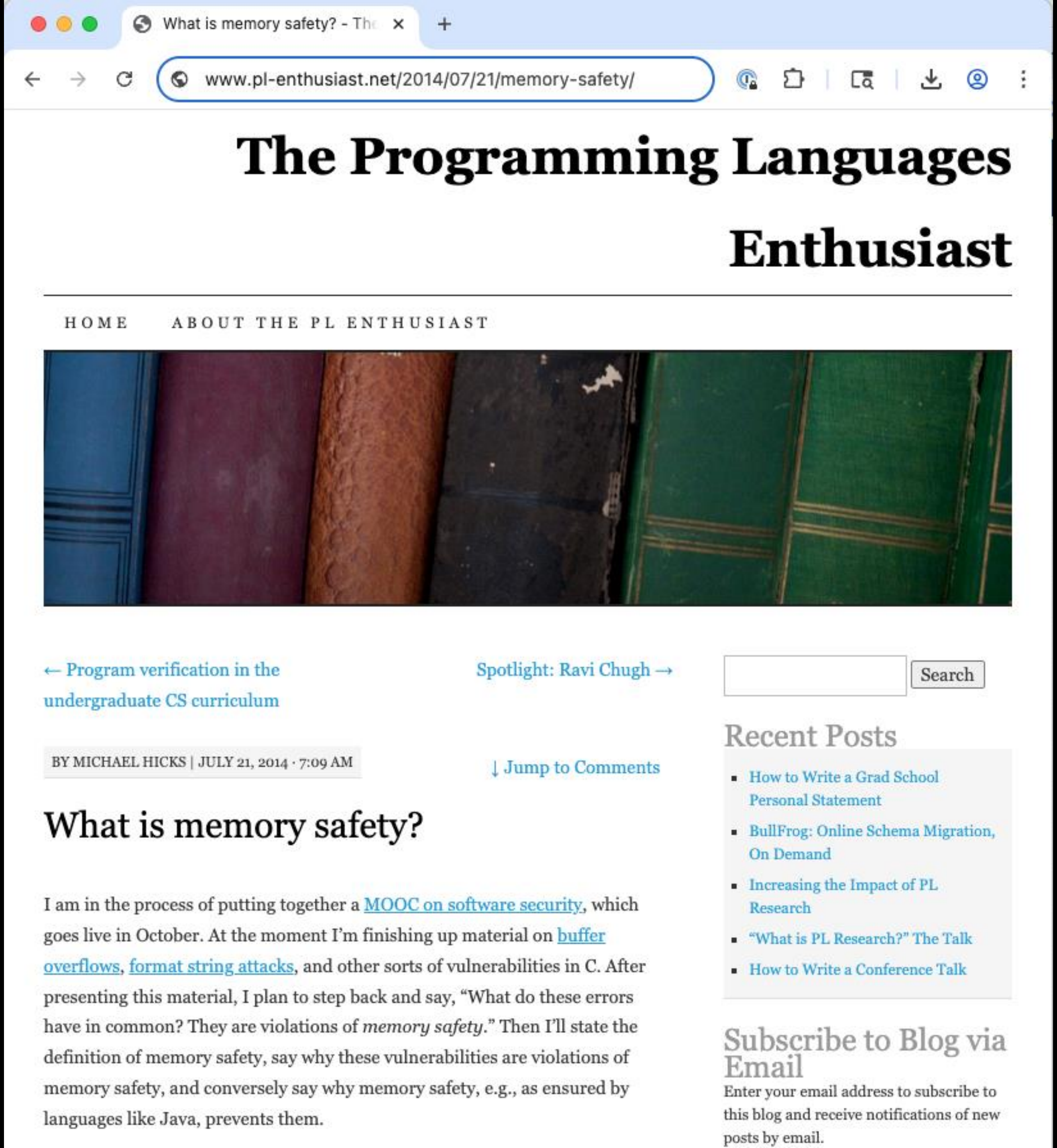
Michael Hicks

How would you answer this question?

In the last decade, has the **security of computer systems**, generally,

- **improved**,
- **declined**, or
- **stayed the same**?

A story of memory (un)safety


What is memory safety? - Th
www.pl-enthusiast.net/2014/07/21/memory-safety/

The Programming Languages Enthusiast

HOME   ABOUT THE PL ENTHUSIAST

← Program verification in the undergraduate CS curriculum

Spotlight: Ravi Chugh →

BY MICHAEL HICKS | JULY 21, 2014 · 7:09 AM

↓ Jump to Comments

What is memory safety?

I am in the process of putting together a MOOC on software security, which goes live in October. At the moment I'm finishing up material on buffer overflows, format string attacks, and other sorts of vulnerabilities in C. After presenting this material, I plan to step back and say, "What do these errors have in common? They are violations of *memory safety*." Then I'll state the definition of memory safety, say why these vulnerabilities are violations of memory safety, and conversely say why memory safety, e.g., as ensured by languages like Java, prevents them.

Recent Posts

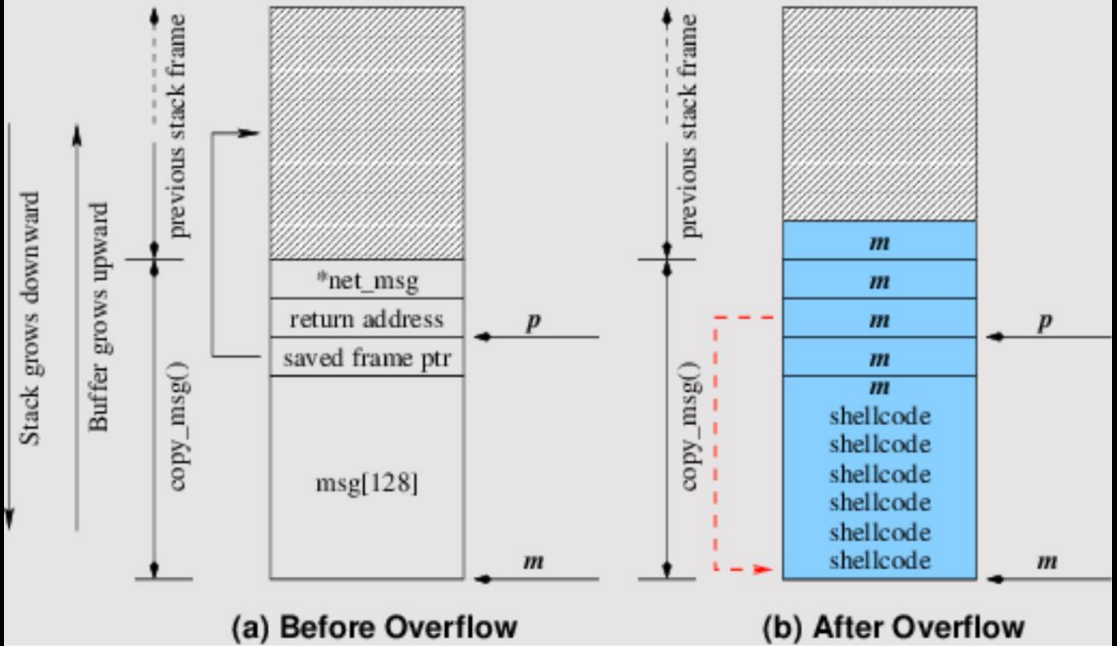
- How to Write a Grad School Personal Statement
- BullFrog: Online Schema Migration, On Demand
- Increasing the Impact of PL Research
- "What is PL Research?" The Talk
- How to Write a Conference Talk

Subscribe to Blog via Email

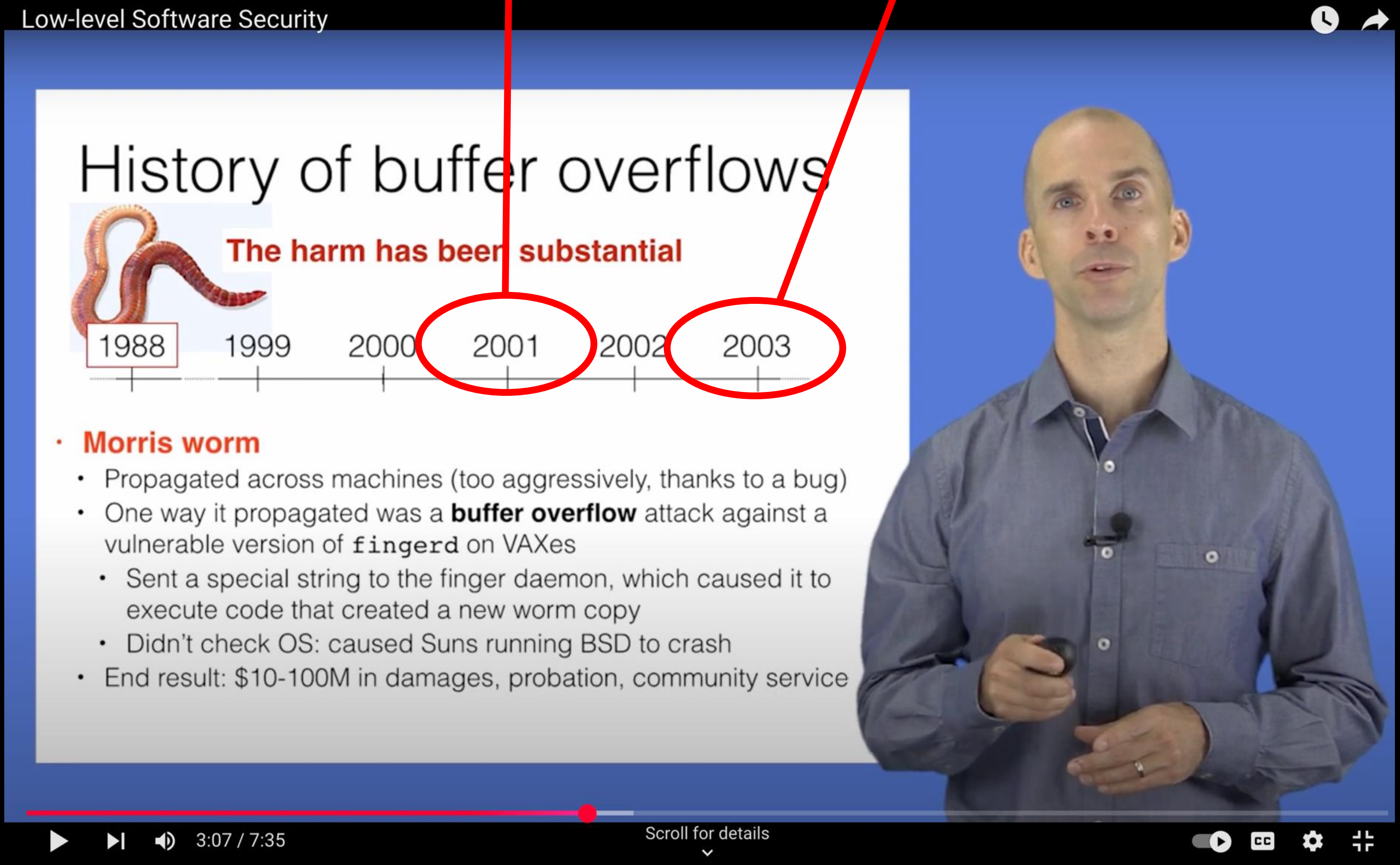
Enter your email address to subscribe to this blog and receive notifications of new posts by email.

Violations of memory safety

- Spatial
    - Buffer overflow (heap or stack, read or write)
- Temporal
    - Use after free
    - Use of uninitialized memory
- Other (maybe)
    - Wild pointer deference (int to pointer, deref)
    - Type confusion (bad cast, deref)


Stack grows downward
Buffer grows upward
previous stack frame
copy_msg()
*net_msg
return address
saved frame ptr
msg[128]
p
m
(a) Before Overflow
previous stack frame
copy_msg()
m
m
m
m
m
shellcode
shellcode
shellcode
shellcode
shellcode
shellcode
p
m
(b) After Overflow

Code Red SQL Slammer



History of buffer overflows

The harm has been substantial

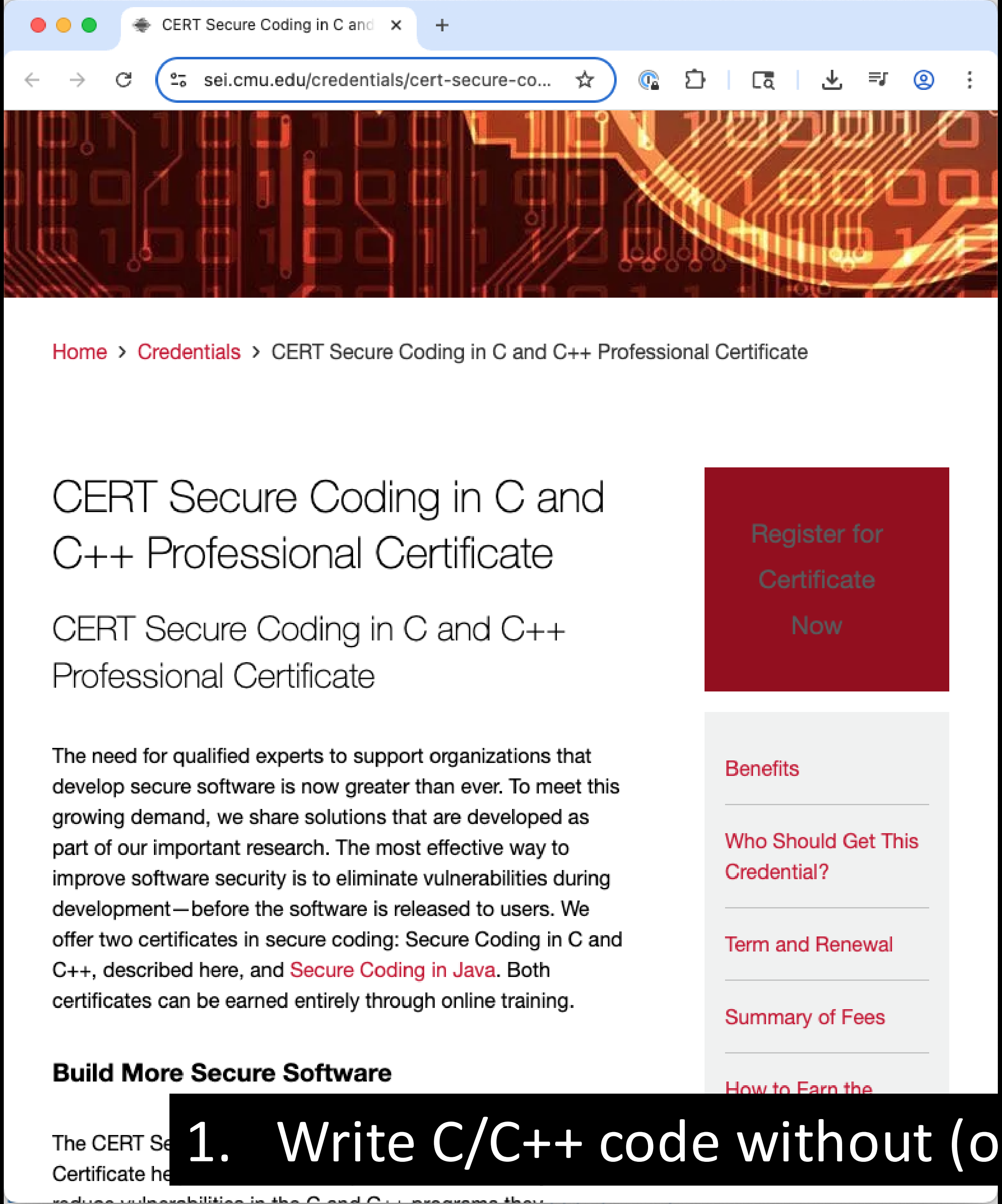
1988 1999 2000 2001 2002 2003

- **Morris worm**
  - Propagated across machines (too aggressively, thanks to a bug)
  - One way it propagated was a **buffer overflow** attack against a vulnerable version of `fingerd` on VAXes
    - Sent a special string to the finger daemon, which caused it to execute code that created a new worm copy
    - Didn't check OS: caused Suns running BSD to crash
  - End result: $10-100M in damages, probation, community service

https://www.youtube.com/watch?v=3BqiTEwz1I0

What to do? Some options

1. Write C/C++ code without (or with fewer of) these bugs in it
2. Write code in a *memory safe* language
3. Leverage compiler and OS-level *mitigations* that
    1. make the bugs more difficult to exploit, and/or
    2. limit the damage/scope of an exploitation



CERT Secure Coding in C and
sei.cmu.edu/credentials/cert-secure-co...

Home › Credentials › CERT Secure Coding in C and C++ Professional Certificate

CERT Secure Coding in C and C++ Professional Certificate

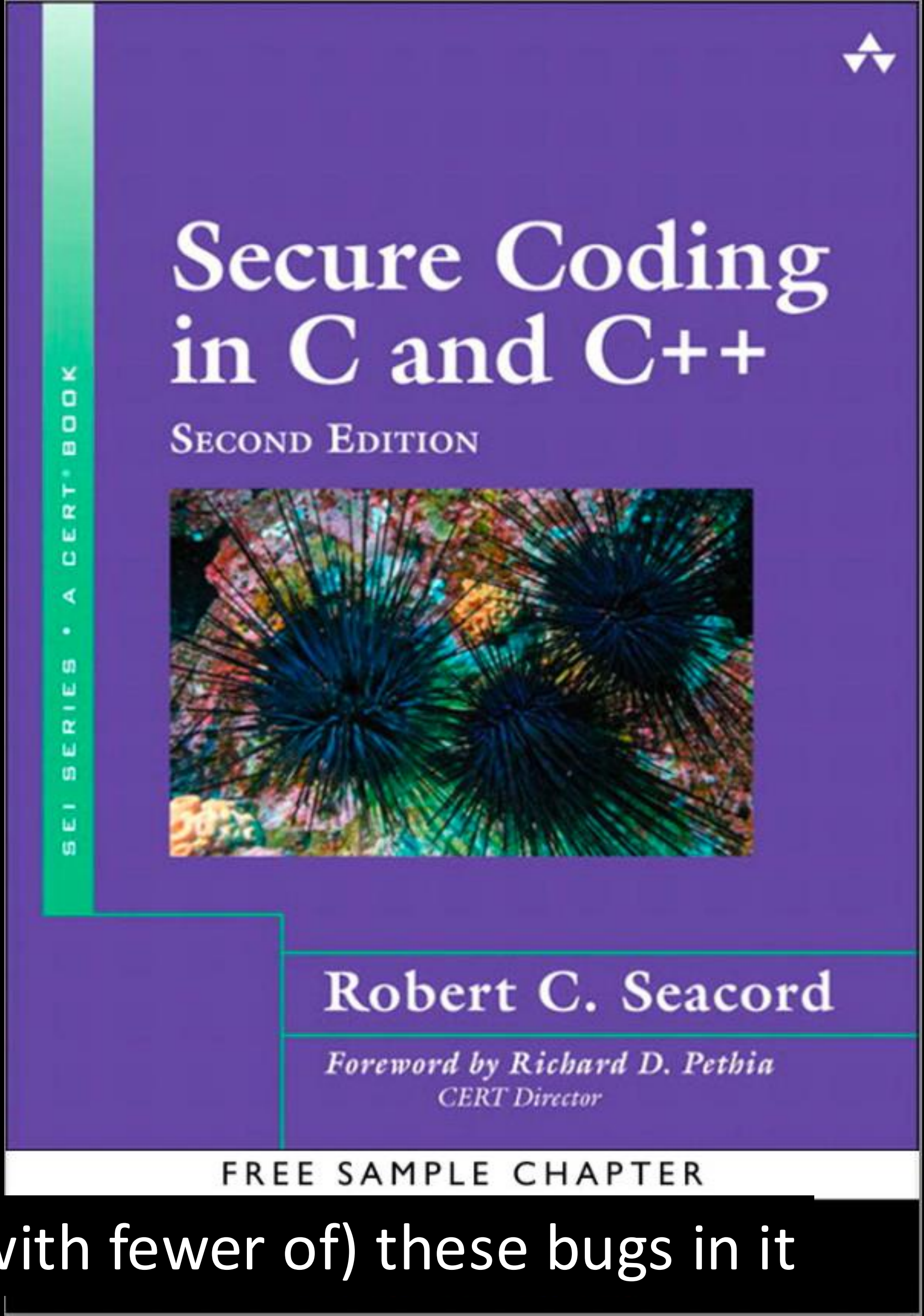
CERT Secure Coding in C and C++ Professional Certificate

The need for qualified experts to support organizations that develop secure software is now greater than ever. To meet this growing demand, we share solutions that are developed as part of our important research. The most effective way to improve software security is to eliminate vulnerabilities during development—before the software is released to users. We offer two certificates in secure coding: Secure Coding in C and C++, described here, and Secure Coding in Java. Both certificates can be earned entirely through online training.

Build More Secure Software

The CERT Se
Certificate he
reduce vulnerabilities in the C and C++ programs they

Register for Certificate Now
Benefits
Who Should Get This Credential?
Term and Renewal
Summary of Fees
How to Earn the



SEI SERIES • A CERT® BOOK

Secure Coding in C and C++

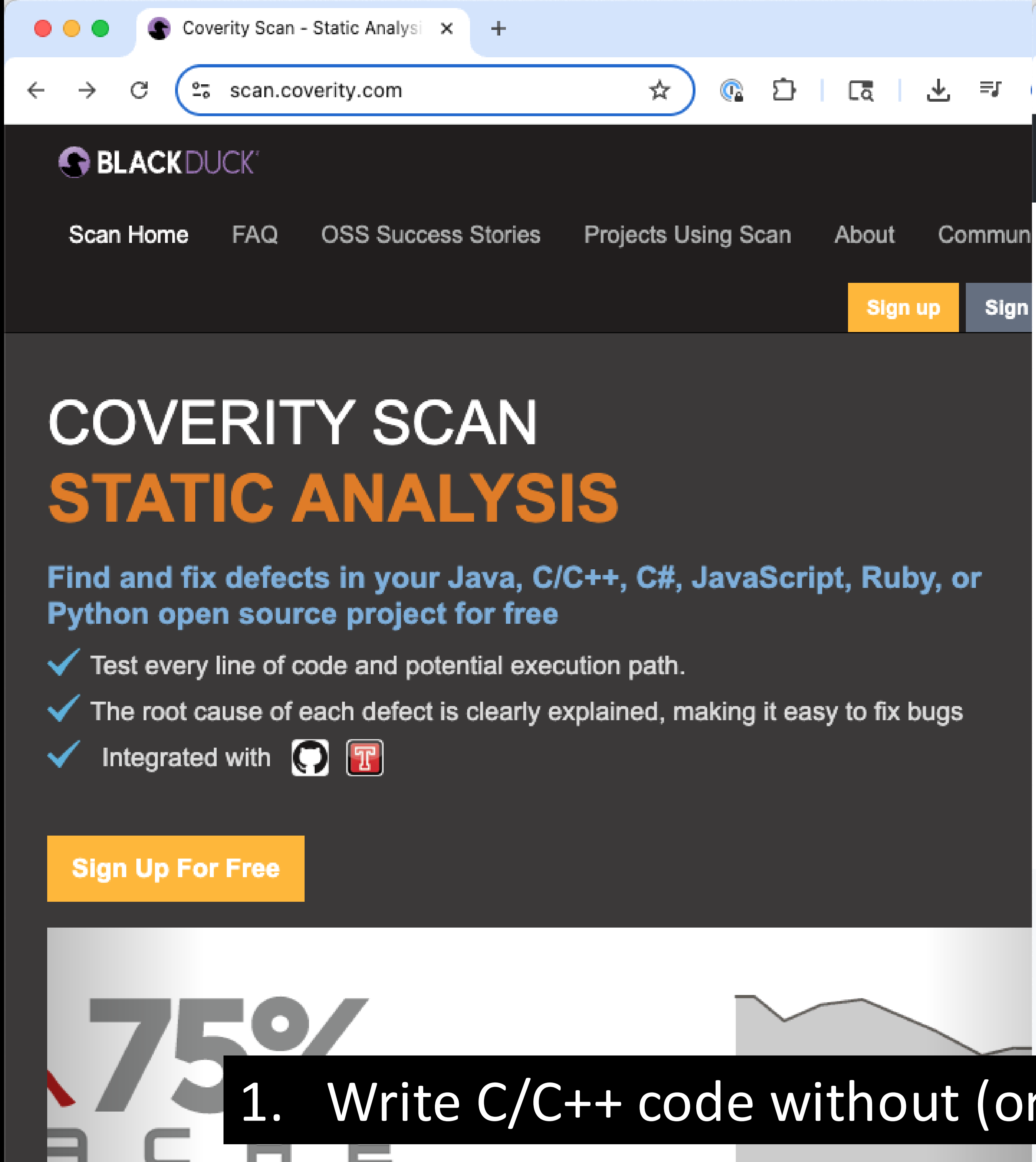
Second Edition

Robert C. Seacord

Foreword by Richard D. Pethia
CERT Director

FREE SAMPLE CHAPTER

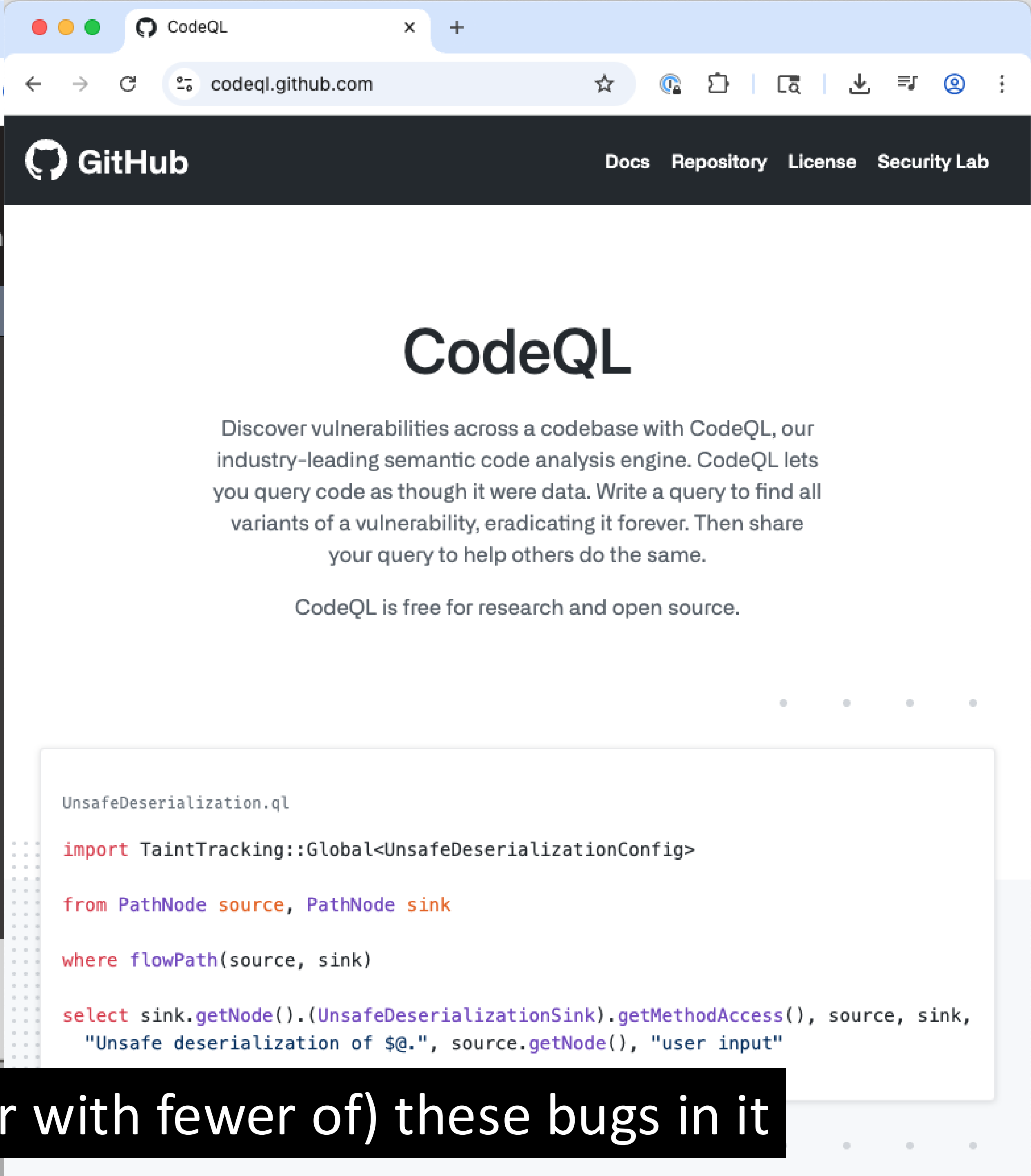
1. Write C/C++ code without (or with fewer of) these bugs in it



COVERITY SCAN
STATIC ANALYSIS

Find and fix defects in your Java, C/C++, C#, JavaScript, Ruby, or Python open source project for free

- ✓ Test every line of code and potential execution path.
- ✓ The root cause of each defect is clearly explained, making it easy to fix bugs
- ✓ Integrated with



CodeQL

Discover vulnerabilities across a codebase with CodeQL, our industry-leading semantic code analysis engine. CodeQL lets you query code as though it were data. Write a query to find all variants of a vulnerability, eradicating it forever. Then share your query to help others do the same.

CodeQL is free for research and open source.

UnsafeDeserialization.ql

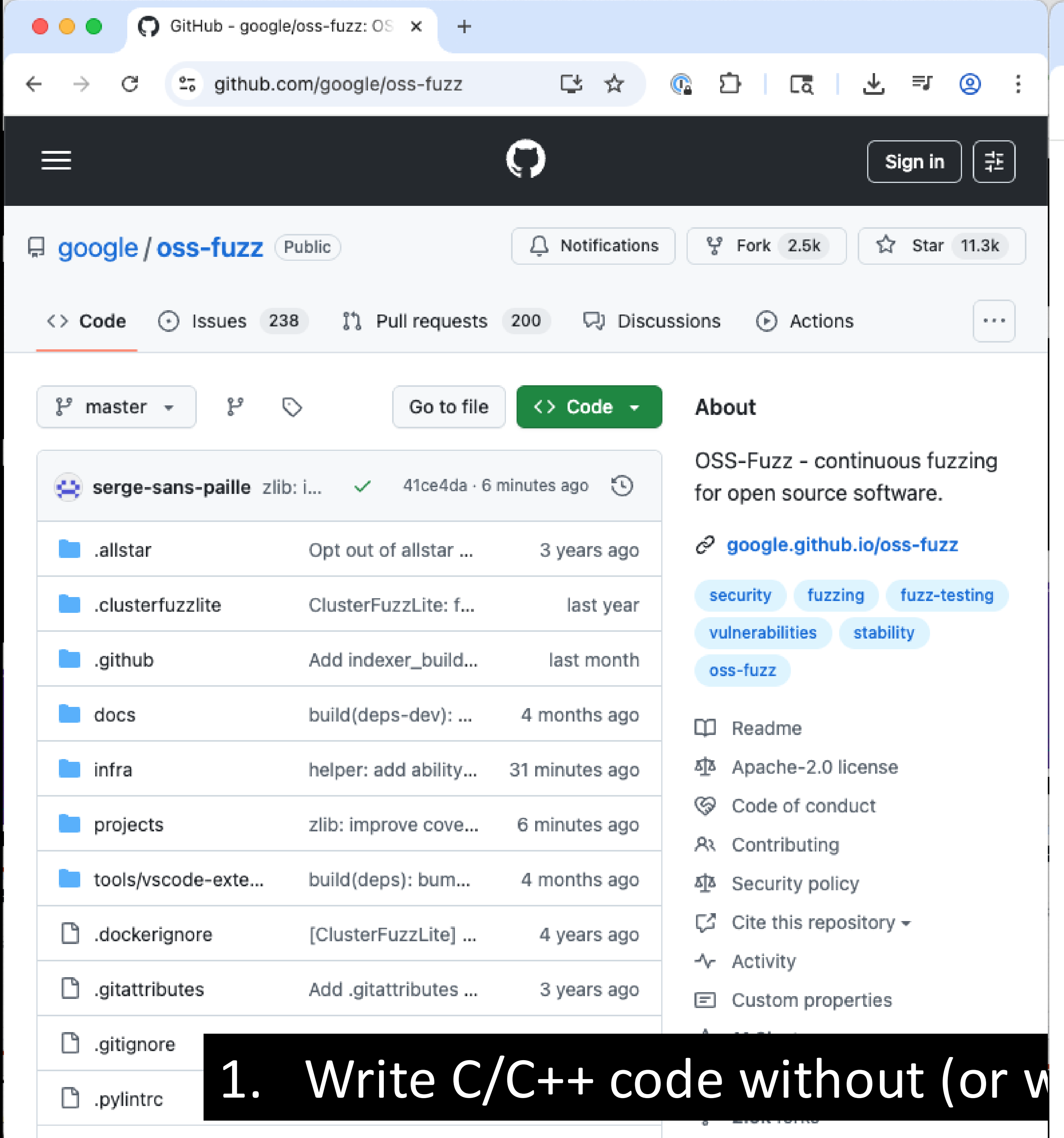
```
import TaintTracking::Global<UnsafeDeserializationConfig>

from PathNode source, PathNode sink

where flowPath(source, sink)

select sink.getNode().(UnsafeDeserializationSink).getMethodAccess(), source, sink,
  "Unsafe deserialization of $@.", source.getNode(), "user input"
```

1. Write C/C++ code without (or with fewer of) these bugs in it



GitHub - google/oss-fuzz: OS

github.com/google/oss-fuzz

Sign in

google / oss-fuzz Public

Notifications Fork 2.5k Star 11.3k

Code Issues 238 Pull requests 200 Discussions Actions

master Go to file Code

serge-sans-paille zlib: i... ✓ 41ce4da · 6 minutes ago

.allstar	Opt out of allstar ...	3 years ago
.clusterfuzzlite	ClusterFuzzLite: f...	last year
.github	Add indexer_build...	last month
docs	build(deps-dev): ...	4 months ago
infra	helper: add ability...	31 minutes ago
projects	zlib: improve cove...	6 minutes ago
tools/vscode-exte...	build(deps): bum...	4 months ago
.dockerignore	[ClusterFuzzLite] ...	4 years ago
.gitattributes	Add .gitattributes ...	3 years ago
.gitignore		
.pylintrc		

About

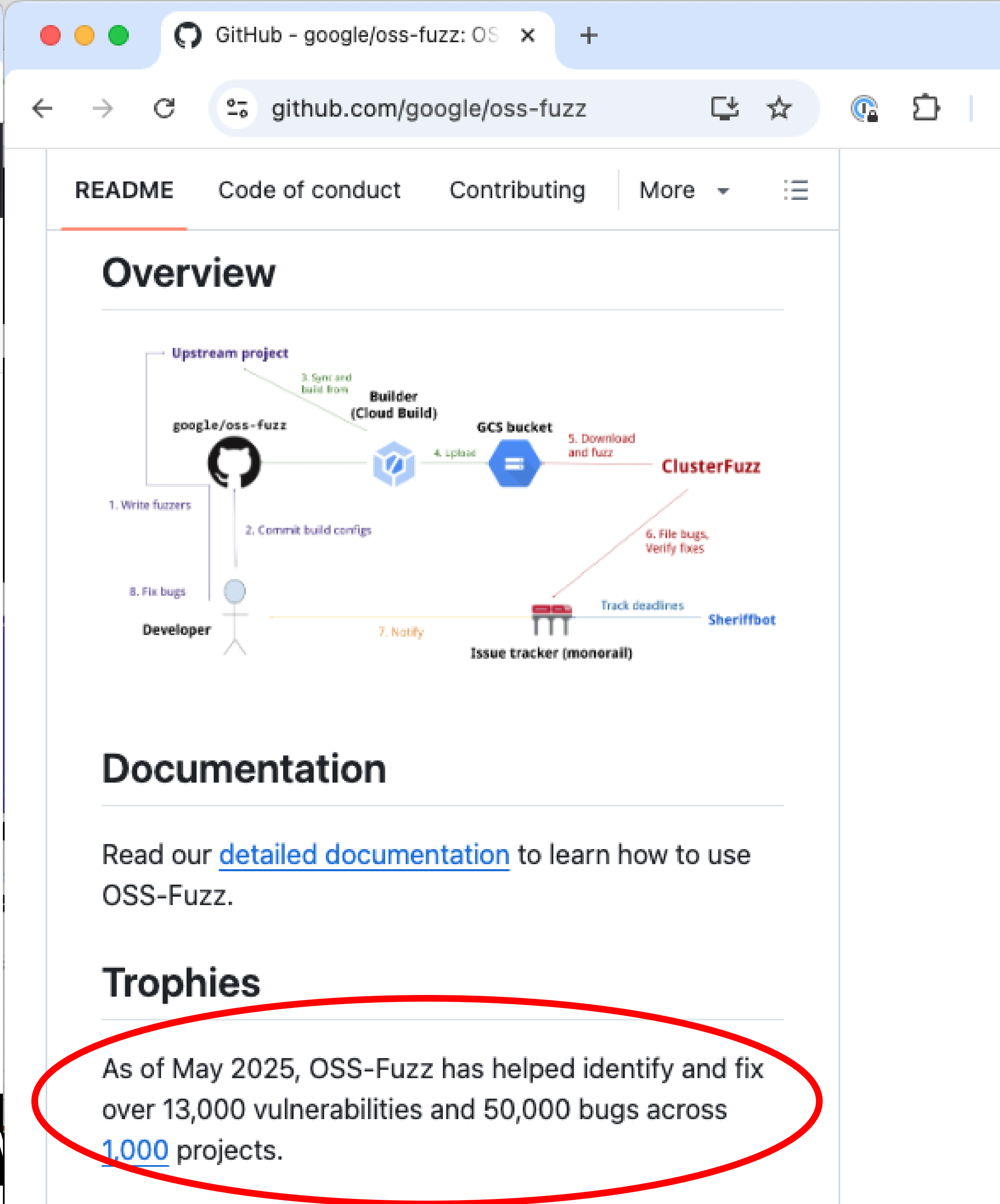
OSS-Fuzz - continuous fuzzing for open source software.

google.github.io/oss-fuzz

security fuzzing fuzz-testing vulnerabilities stability oss-fuzz

- Readme
- Apache-2.0 license
- Code of conduct
- Contributing
- Security policy
- Cite this repository
- Activity
- Custom properties

1. Write C/C++ code without (or w



GitHub - google/oss-fuzz: OS

github.com/google/oss-fuzz

README Code of conduct Contributing More

Overview

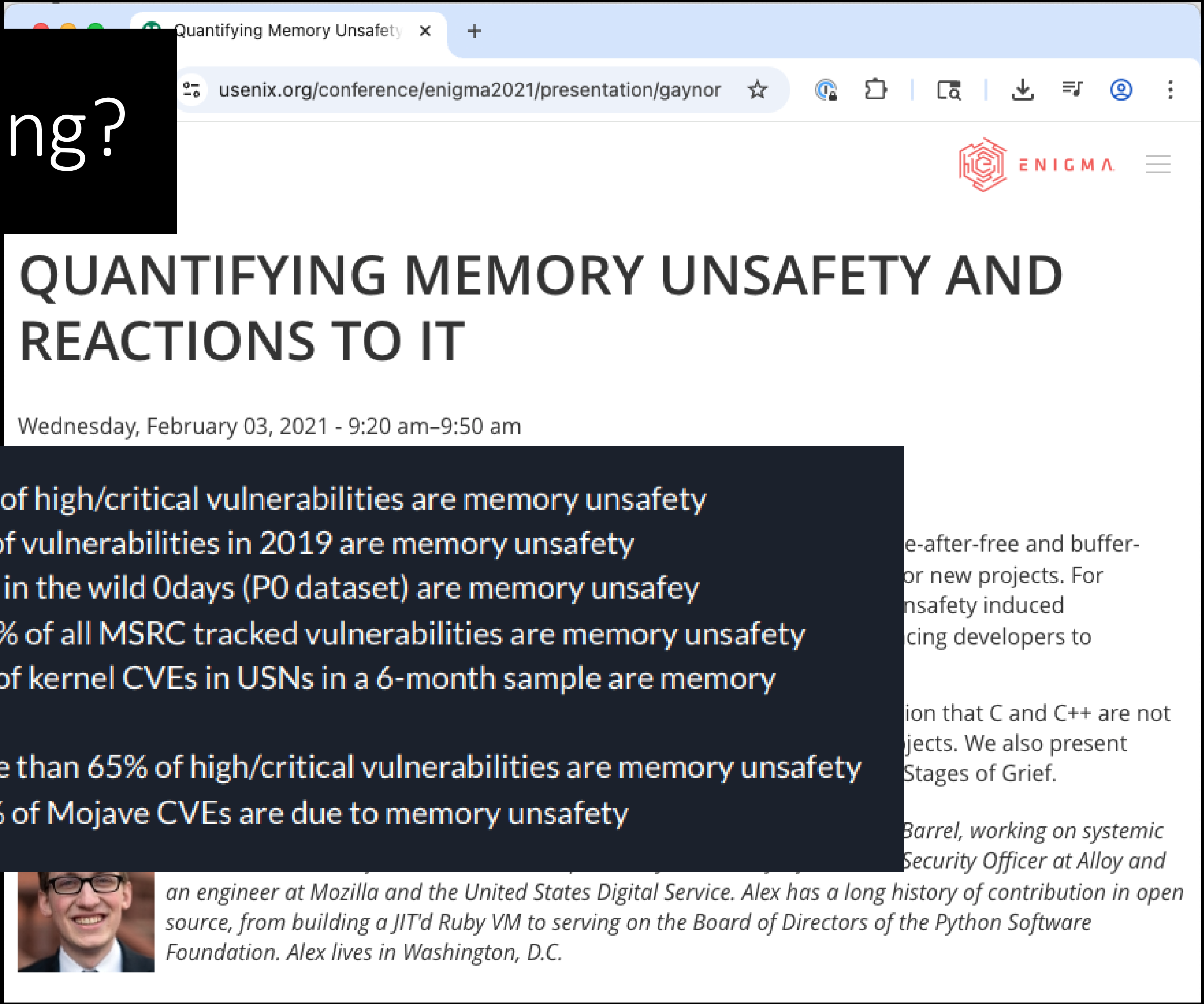
Documentation

Read our detailed documentation to learn how to use OSS-Fuzz.

Trophies

As of May 2025, OSS-Fuzz has helped identify and fix over 13,000 vulnerabilities and 50,000 bugs across 1,000 projects.

How’s that going?



Quantifying Memory Unsafety
usenix.org/conference/enigma2021/presentation/gaynor
ENIGMA

QUANTIFYING MEMORY UNSAFETY AND REACTIONS TO IT

Wednesday, February 03, 2021 - 9:20 am–9:50 am

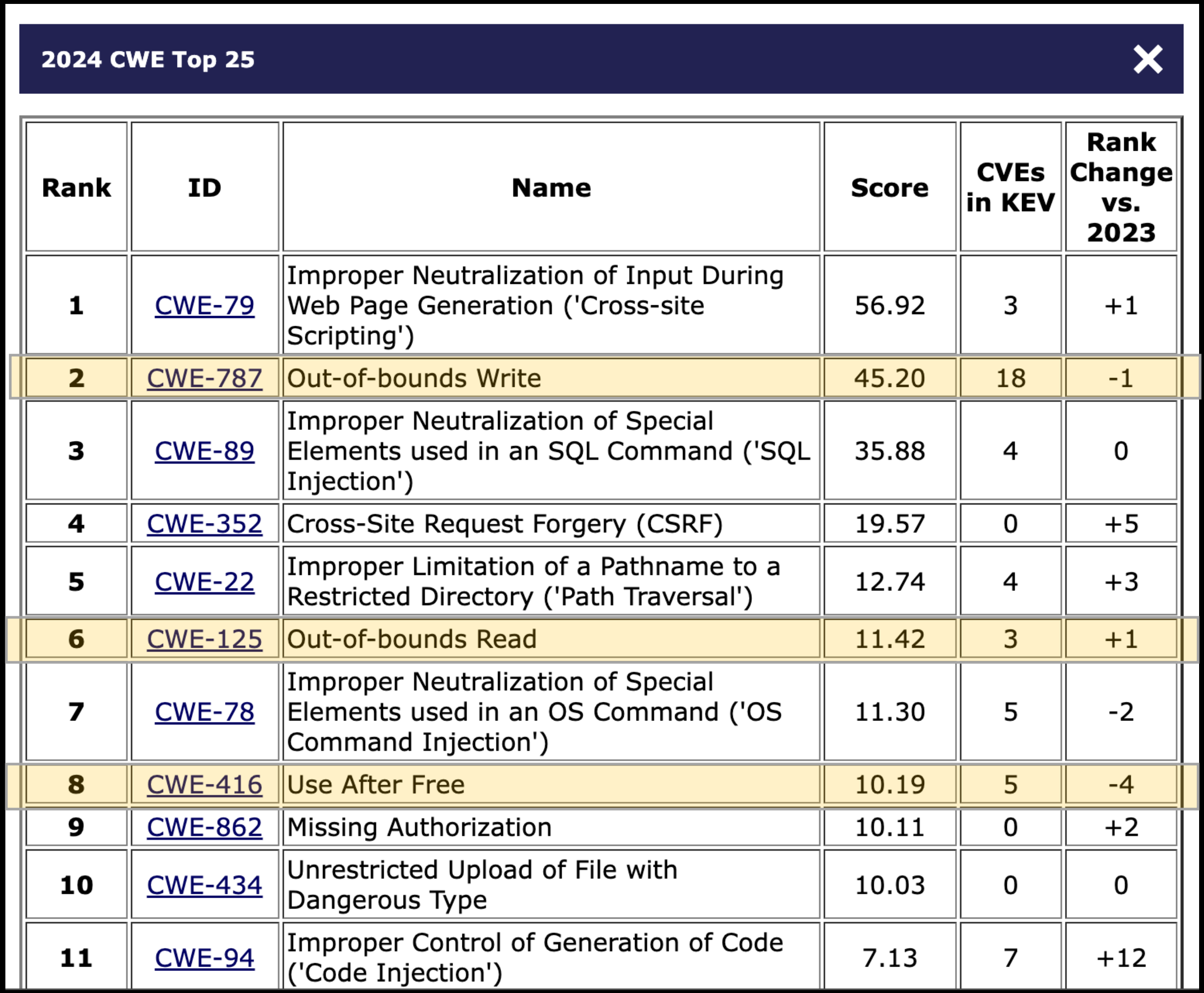
e-after-free and buffer-
or new projects. For
nsafety induced
cing developers to

ion that C and C++ are not
jects. We also present
Stages of Grief.

Barrel, working on systemic
Security Officer at Alloy and
an engineer at Mozilla and the United States Digital Service. Alex has a long history of contribution in open source, from building a JIT'd Ruby VM to serving on the Board of Directors of the Python Software Foundation. Alex lives in Washington, D.C.

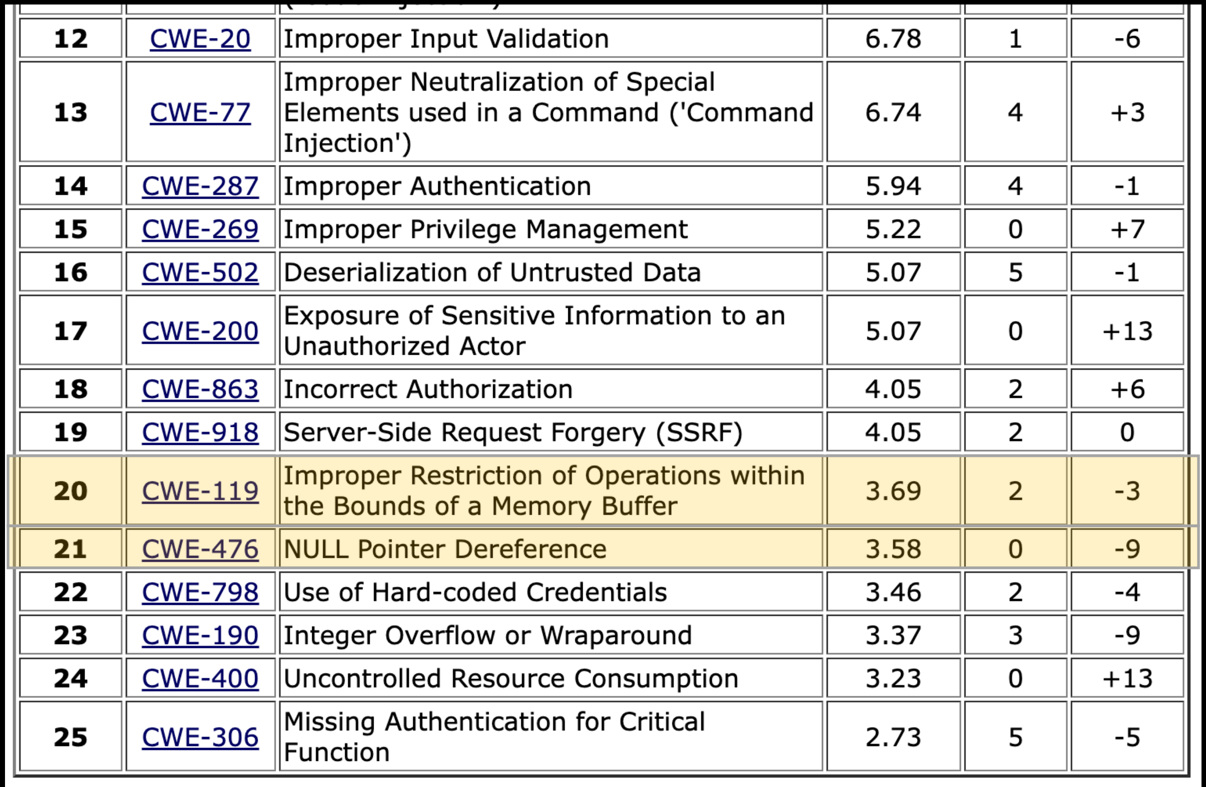
- **Chrome**: 70% of high/critical vulnerabilities are memory unsafety
- **Firefox:** 72% of vulnerabilities in 2019 are memory unsafety
- **0days**: 81% of in the wild 0days (P0 dataset) are memory unsafey
- **Microsoft**: 70% of all MSRC tracked vulnerabilities are memory unsafety
- **Ubuntu**: 65% of kernel CVEs in USNs in a 6-month sample are memory unsafety
- **Android**: More than 65% of high/critical vulnerabilities are memory unsafety
- **macOS**: 71.5% of Mojave CVEs are due to memory unsafety

How’s that going?



2024 CWE Top 25

Classic memory-safety vulnerabilities



Rank	ID	Name	Score	CVEs in KEV	Rank Change vs. 2023
1	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	56.92	3	+1
2	CWE-787	Out-of-bounds Write	45.20	18	-1
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	35.88	4	0
4	CWE-352	Cross-Site Request Forgery (CSRF)	19.57	0	+5
5	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	12.74	4	+3
6	CWE-125	Out-of-bounds Read	11.42	3	+1
7	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	11.30	5	-2
8	CWE-416	Use After Free	10.19	5	-4
9	CWE-862	Missing Authorization	10.11	0	+2
10	CWE-434	Unrestricted Upload of File with Dangerous Type	10.03	0	0
11	CWE-94	Improper Control of Generation of Code ('Code Injection')	7.13	7	+12
12	CWE-20	Improper Input Validation	6.78	1	-6
13	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	6.74	4	+3
14	CWE-287	Improper Authentication	5.94	4	-1
15	CWE-269	Improper Privilege Management	5.22	0	+7
16	CWE-502	Deserialization of Untrusted Data	5.07	5	-1
17	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	5.07	0	+13
18	CWE-863	Incorrect Authorization	4.05	2	+6
19	CWE-918	Server-Side Request Forgery (SSRF)	4.05	2	0
20	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	3.69	2	-3
21	CWE-476	NULL Pointer Dereference	3.58	0	-9
22	CWE-798	Use of Hard-coded Credentials	3.46	2	-4
23	CWE-190	Integer Overflow or Wraparound	3.37	3	-9
24	CWE-400	Uncontrolled Resource Consumption	3.23	0	+13
25	CWE-306	Missing Authentication for Critical Function	2.73	5	-5

https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html

What to do? Some options

1. Write C/C++ code without (or with fewer of) these bugs in it
2. Write code in a *memory safe* language
3. Leverage compiler and OS-level *mitigations* that
   1. make the bugs more difficult to exploit, and/or
   2. limit the damage/scope of an exploitation

2. Write code in a *memory safe* language

Memory safe:

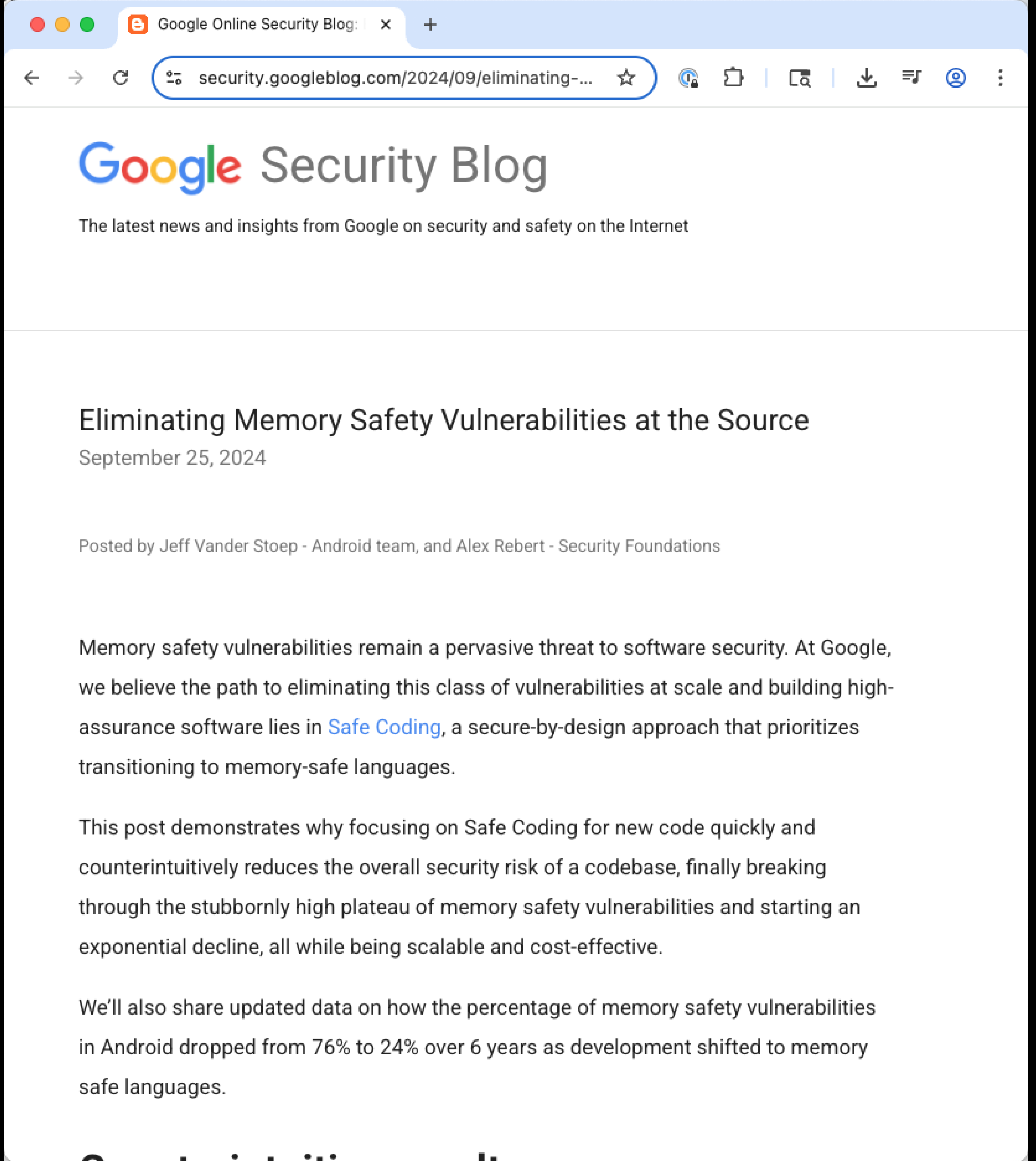
- Rust
- Swift
- Go
- Haskell
- Python
- Etc.

Memory unsafe:

- C
- C++
- Assembly

How’s that going?

the percentage of memory safety vulnerabilities in Android dropped from 76% to 24% over 6 years as development shifted to memory safe languages


Google Online Security Blog:
security.googleblog.com/2024/09/eliminating-...

Google Security Blog

The latest news and insights from Google on security and safety on the Internet

Eliminating Memory Safety Vulnerabilities at the Source

September 25, 2024

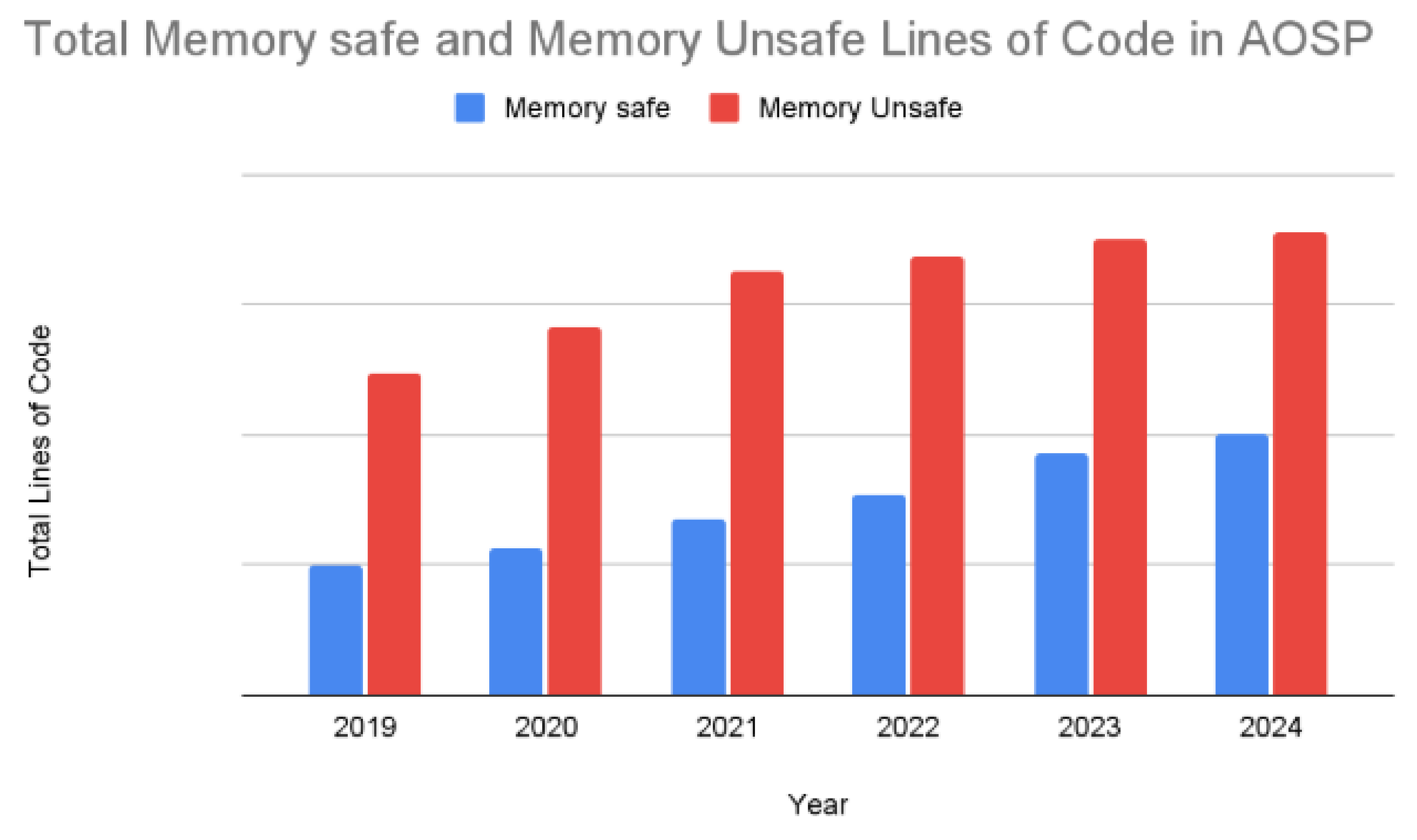
Posted by Jeff Vander Stoep - Android team, and Alex Rebert - Security Foundations

Memory safety vulnerabilities remain a pervasive threat to software security. At Google, we believe the path to eliminating this class of vulnerabilities at scale and building high-assurance software lies in Safe Coding, a secure-by-design approach that prioritizes transitioning to memory-safe languages.

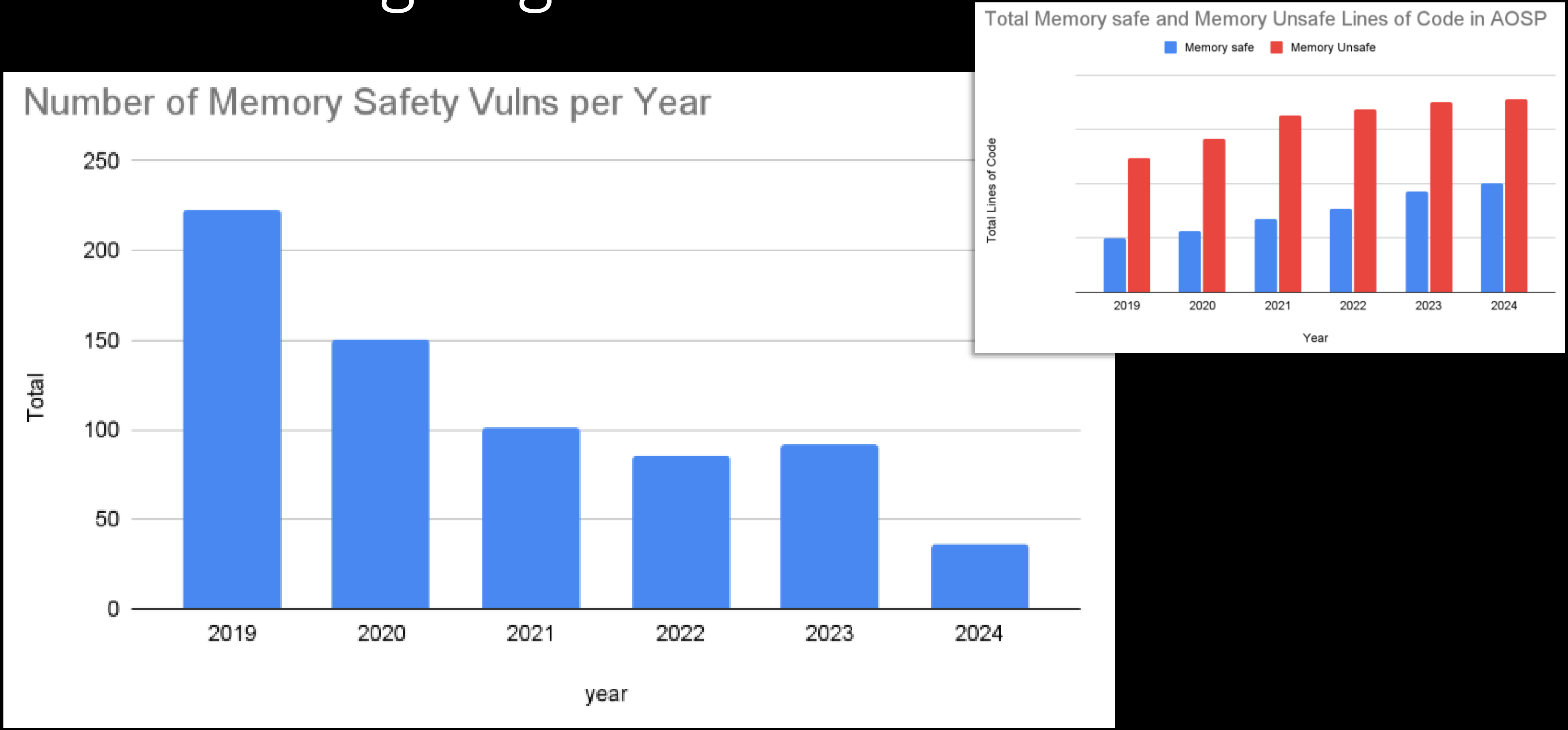
This post demonstrates why focusing on Safe Coding for new code quickly and counterintuitively reduces the overall security risk of a codebase, finally breaking through the stubbornly high plateau of memory safety vulnerabilities and starting an exponential decline, all while being scalable and cost-effective.

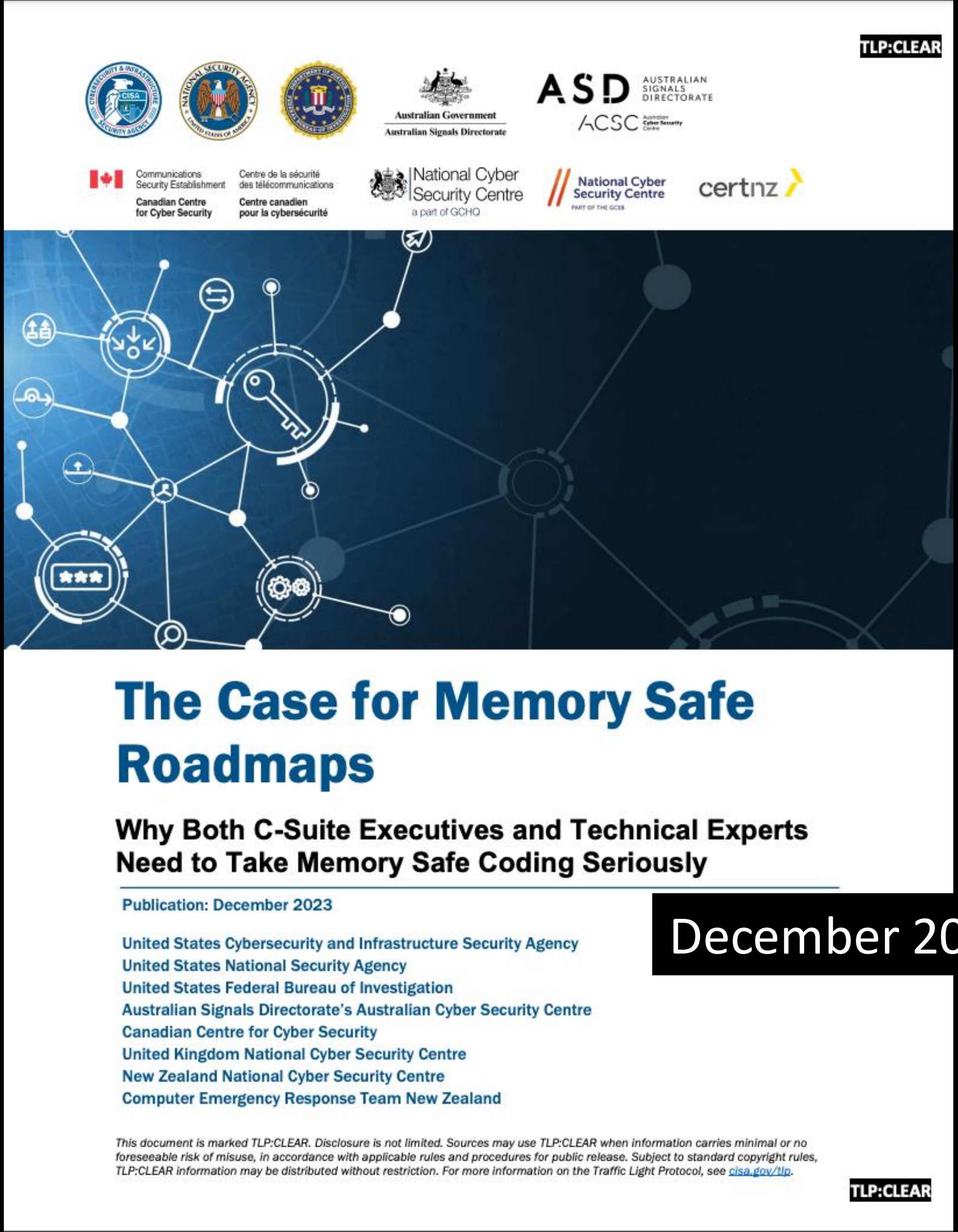
We’ll also share updated data on how the percentage of memory safety vulnerabilities in Android dropped from 76% to 24% over 6 years as development shifted to memory safe languages.

How's that going?



How’s that going?





TLP:CLEAR

The Case for Memory Safe Roadmaps

Why Both C-Suite Executives and Technical Experts Need to Take Memory Safe Coding Seriously

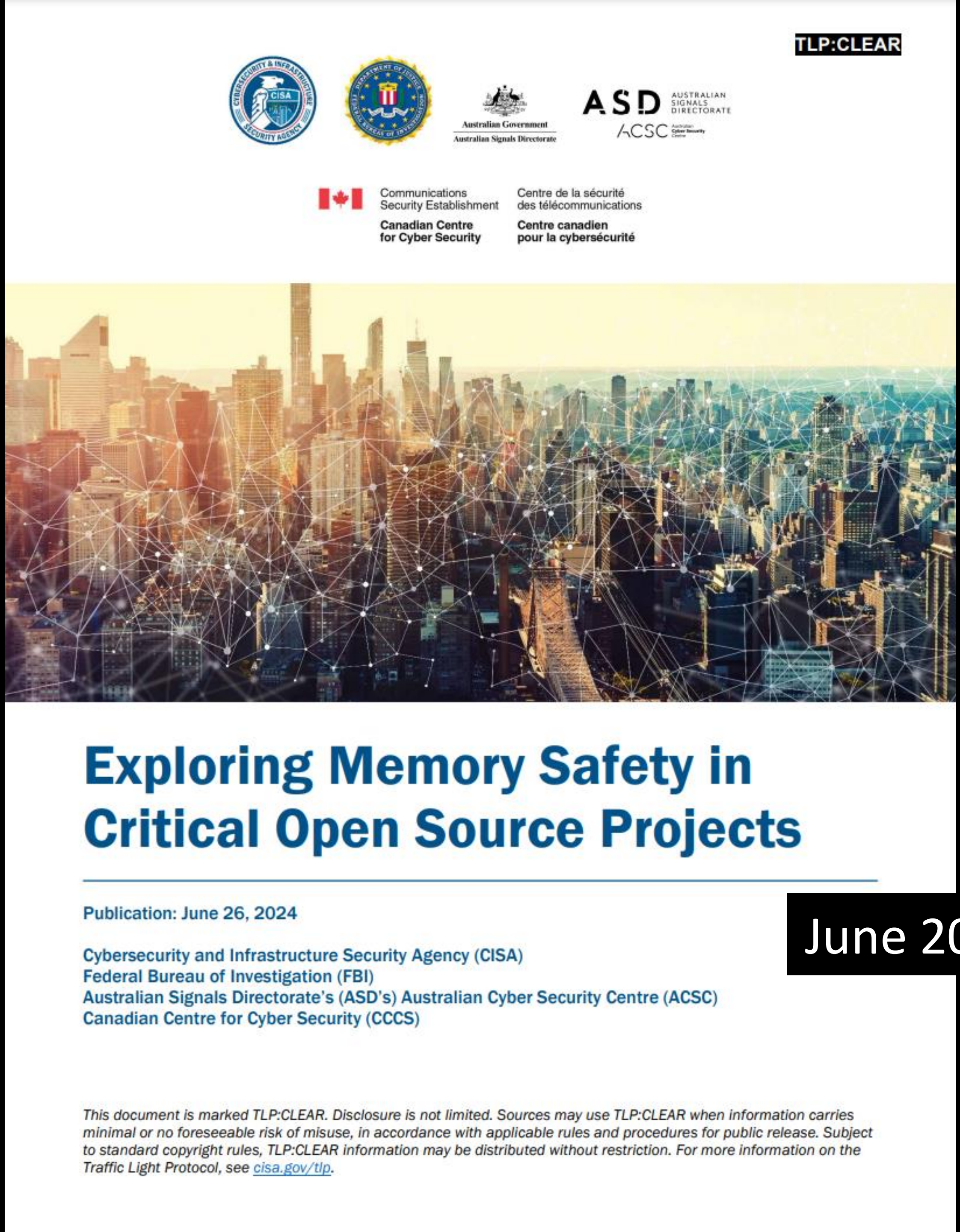
Publication: December 2023

United States Cybersecurity and Infrastructure Security Agency
United States National Security Agency
United States Federal Bureau of Investigation
Australian Signals Directorate's Australian Cyber Security Centre
Canadian Centre for Cyber Security
United Kingdom National Cyber Security Centre
New Zealand National Cyber Security Centre
Computer Emergency Response Team New Zealand

This document is marked TLP:CLEAR. Disclosure is not limited. Sources may use TLP:CLEAR when information carries minimal or no foreseeable risk of misuse, in accordance with applicable rules and procedures for public release. Subject to standard copyright rules, TLP:CLEAR information may be distributed without restriction. For more information on the Traffic Light Protocol, see cisa.gov/tlp.

TLP:CLEAR

December 2023



TLP:CLEAR

Exploring Memory Safety in Critical Open Source Projects

Publication: June 26, 2024

Cybersecurity and Infrastructure Security Agency (CISA)
Federal Bureau of Investigation (FBI)
Australian Signals Directorate's (ASD's) Australian Cyber Security Centre (ACSC)
Canadian Centre for Cyber Security (CCCS)

This document is marked TLP:CLEAR. Disclosure is not limited. Sources may use TLP:CLEAR when information carries minimal or no foreseeable risk of misuse, in accordance with applicable rules and procedures for public release. Subject to standard copyright rules, TLP:CLEAR information may be distributed without restriction. For more information on the Traffic Light Protocol, see cisa.gov/tlp.

June 2024

What to do? Some options

1. Write C/C++ code without (or with fewer of) these bugs in it
2. Write code in a *memory safe* language
3. Leverage compiler and OS-level *mitigations* that
   1. make the bugs more difficult to exploit, and/or
   2. limit the damage/scope of an exploitation

3. Leverage compiler and OS-level *mitigations* that
    1. make the bugs more difficult to exploit, and/or
    2. limit the damage/scope of an exploitation

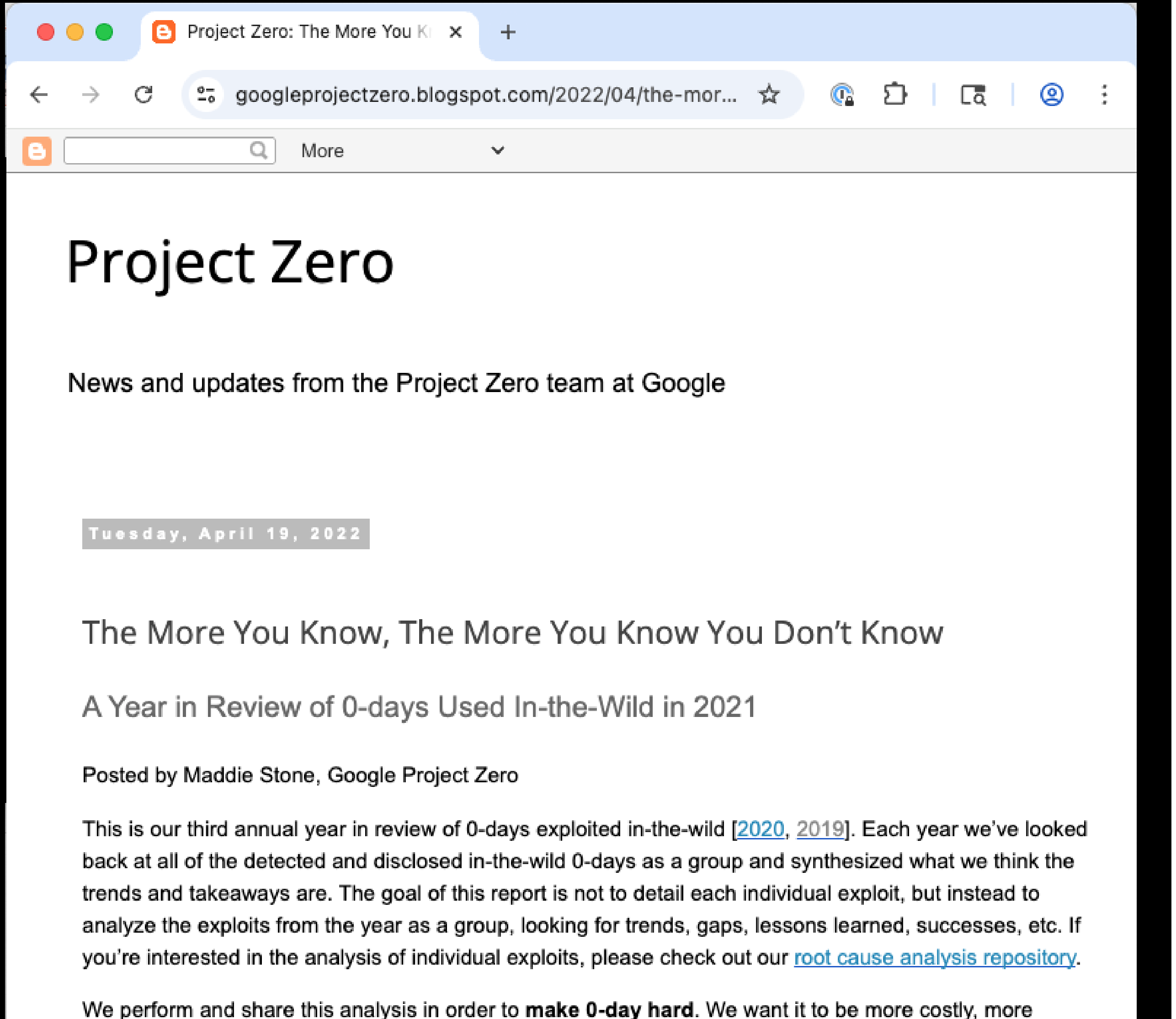
Challenge exploitability:

- Stack canaries
- Address-space layout randomization (ASLR)
- “write xor execute” (W⊕X)
- Control-flow integrity (CFI)
- Etc.

Limit damage:

- Process-level isolation
- Within-process compartments (eg., RLbox)
- Externally enforced access control

How’s that going?


Project Zero: The More You K
googleprojectzero.blogspot.com/2022/04/the-mor...
More

Project Zero

News and updates from the Project Zero team at Google

Tuesday, April 19, 2022

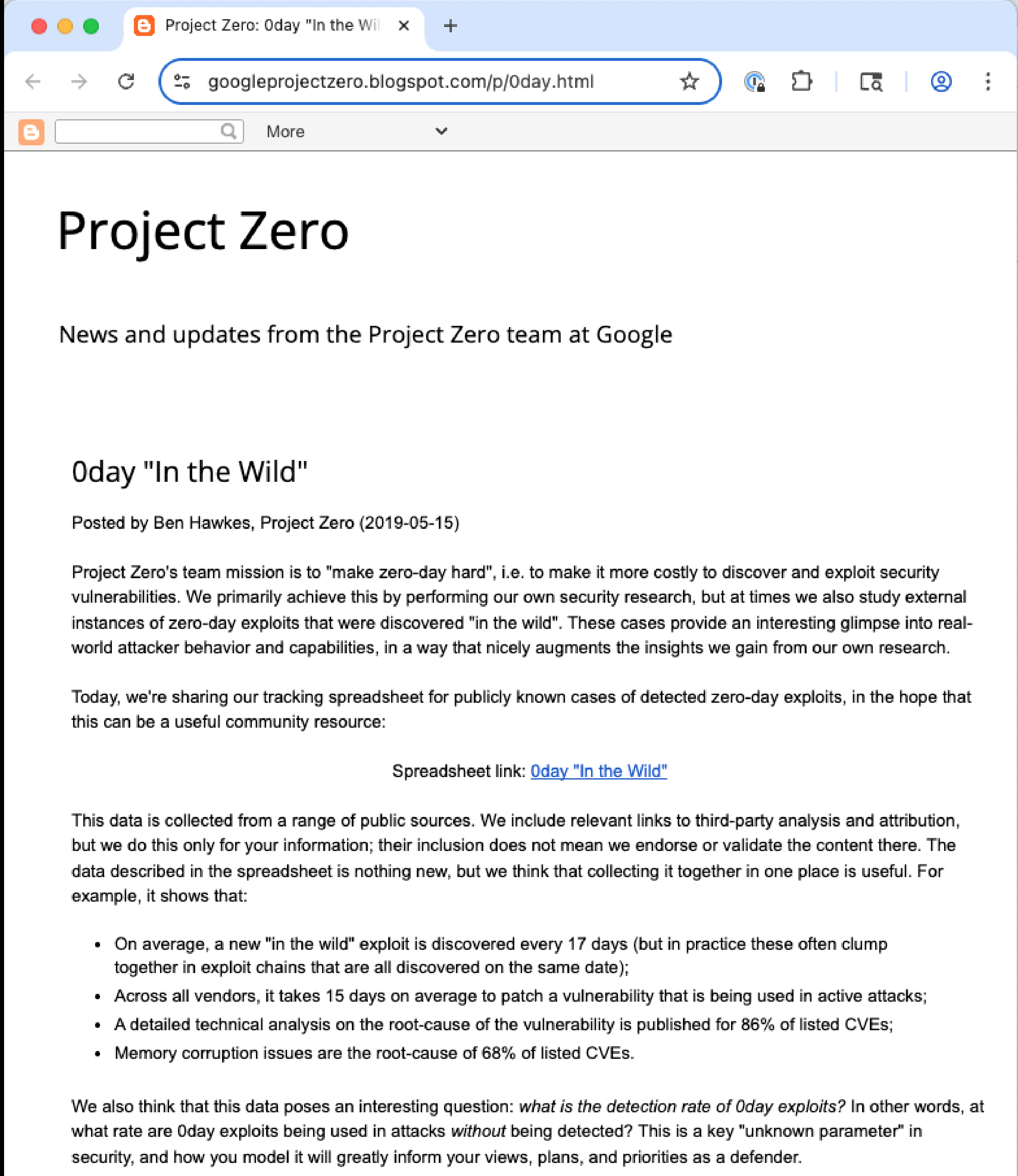
The More You Know, The More You Know You Don’t Know

A Year in Review of 0-days Used In-the-Wild in 2021

Posted by Maddie Stone, Google Project Zero

This is our third annual year in review of 0-days exploited in-the-wild [2020, 2019]. Each year we’ve looked back at all of the detected and disclosed in-the-wild 0-days as a group and synthesized what we think the trends and takeaways are. The goal of this report is not to detail each individual exploit, but instead to analyze the exploits from the year as a group, looking for trends, gaps, lessons learned, successes, etc. If you’re interested in the analysis of individual exploits, please check out our root cause analysis repository.

We perform and share this analysis in order to **make 0-day hard**. We want it to be more costly, more


Project Zero: 0day "In the Wil
googleprojectzero.blogspot.com/p/0day.html
More

Project Zero

News and updates from the Project Zero team at Google

0day "In the Wild"

Posted by Ben Hawkes, Project Zero (2019-05-15)

Project Zero's team mission is to "make zero-day hard", i.e. to make it more costly to discover and exploit security vulnerabilities. We primarily achieve this by performing our own security research, but at times we also study external instances of zero-day exploits that were discovered "in the wild". These cases provide an interesting glimpse into real-world attacker behavior and capabilities, in a way that nicely augments the insights we gain from our own research.

Today, we're sharing our tracking spreadsheet for publicly known cases of detected zero-day exploits, in the hope that this can be a useful community resource:

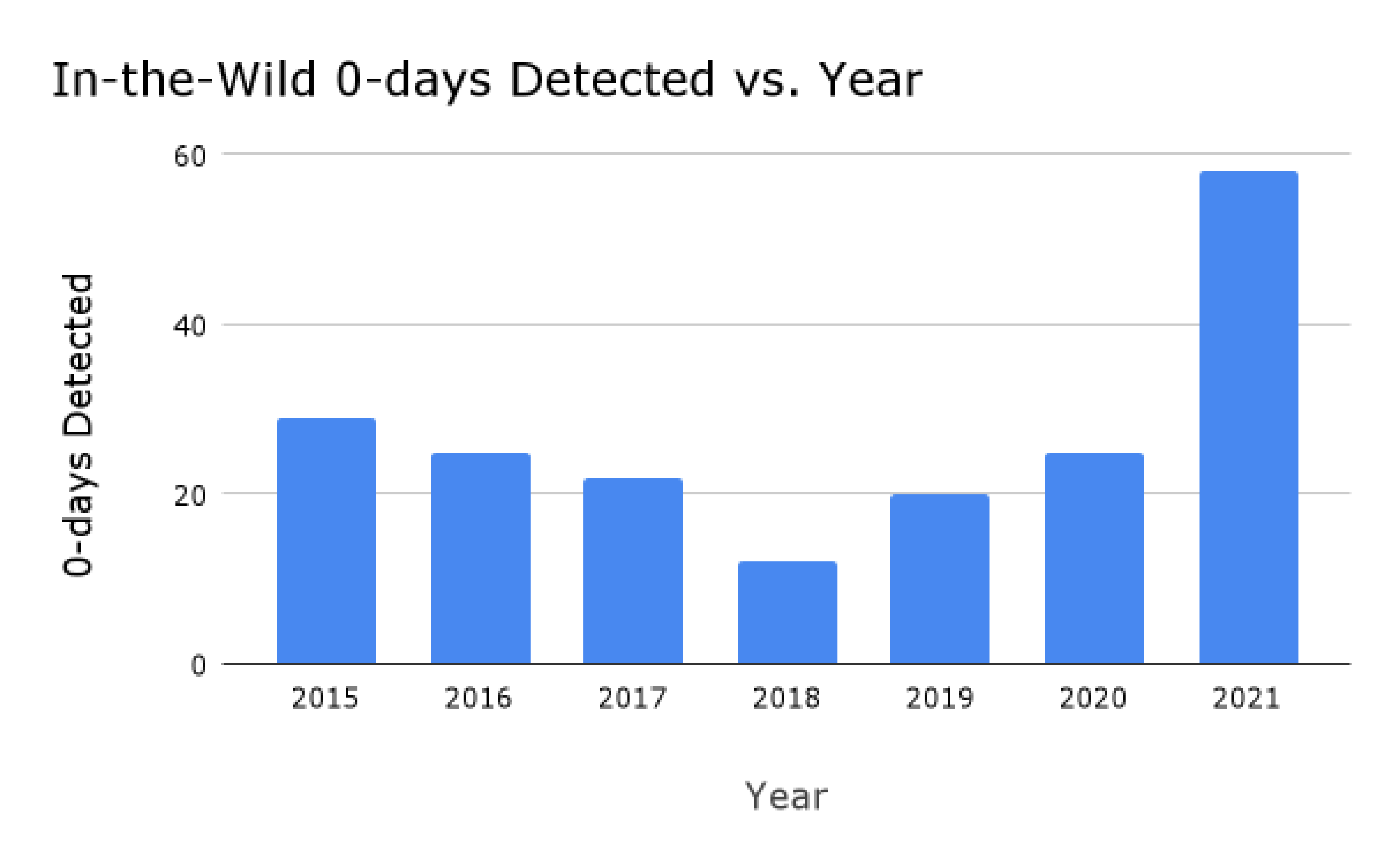
Spreadsheet link: 0day "In the Wild"

This data is collected from a range of public sources. We include relevant links to third-party analysis and attribution, but we do this only for your information; their inclusion does not mean we endorse or validate the content there. The data described in the spreadsheet is nothing new, but we think that collecting it together in one place is useful. For example, it shows that:

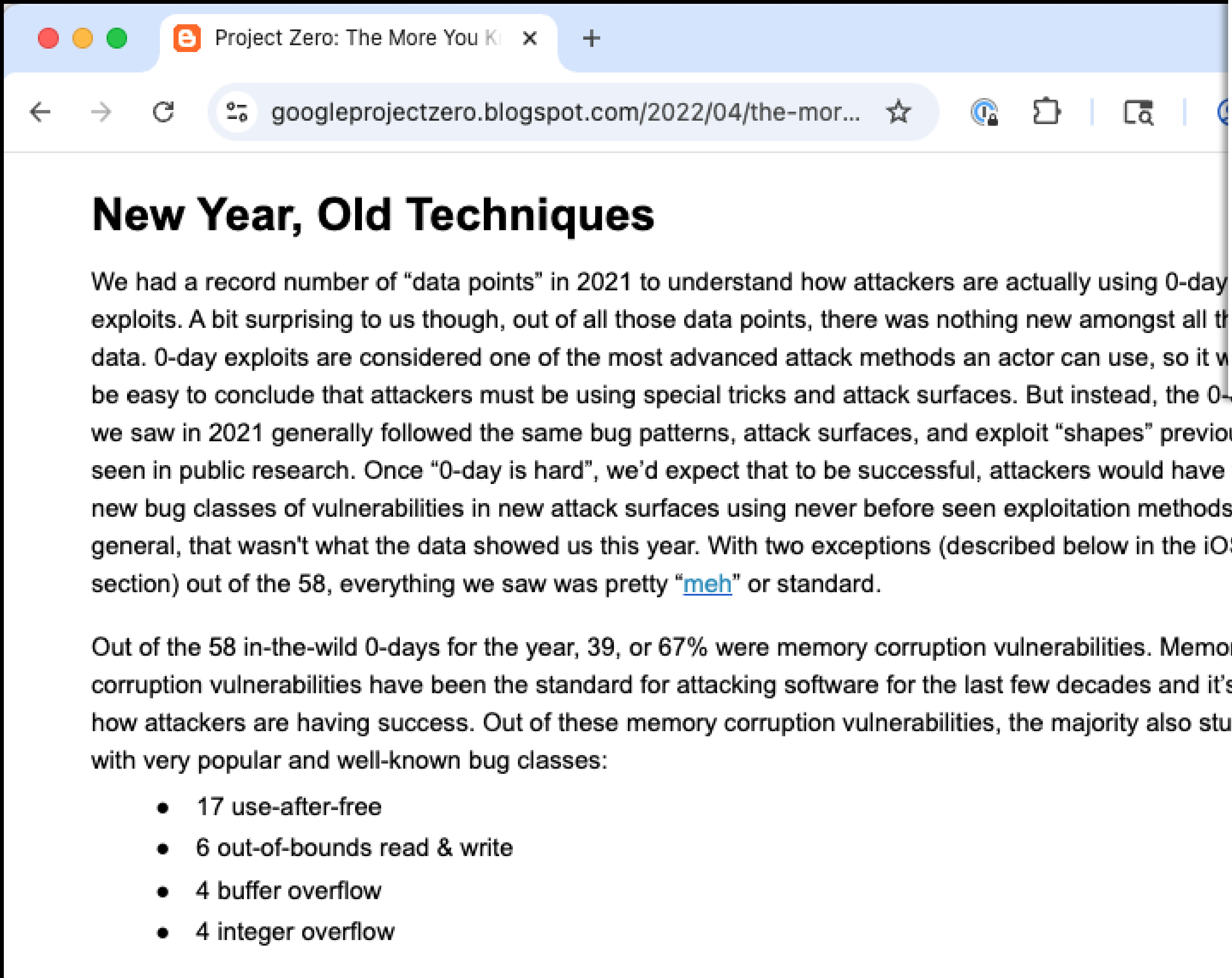
- On average, a new "in the wild" exploit is discovered every 17 days (but in practice these often clump together in exploit chains that are all discovered on the same date);
- Across all vendors, it takes 15 days on average to patch a vulnerability that is being used in active attacks;
- A detailed technical analysis on the root-cause of the vulnerability is published for 86% of listed CVEs;
- Memory corruption issues are the root-cause of 68% of listed CVEs.

We also think that this data poses an interesting question: *what is the detection rate of 0day exploits?* In other words, at what rate are 0day exploits being used in attacks *without* being detected? This is a key "unknown parameter" in security, and how you model it will greatly inform your views, plans, and priorities as a defender.

How’s that going?



How’s that going?

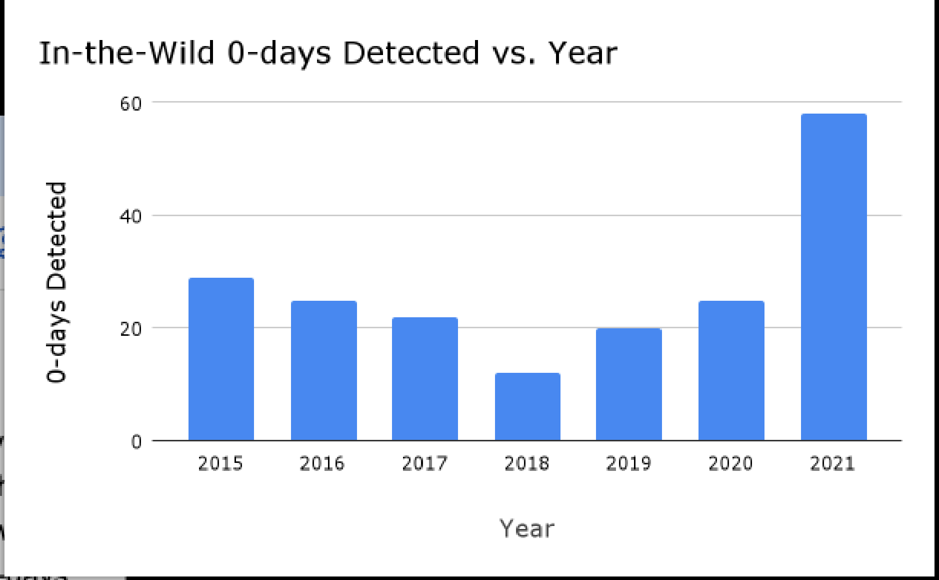

Project Zero: The More You K
googleprojectzero.blogspot.com/2022/04/the-mor...

New Year, Old Techniques

We had a record number of “data points” in 2021 to understand how attackers are actually using 0-day exploits. A bit surprising to us though, out of all those data points, there was nothing new amongst all th data. 0-day exploits are considered one of the most advanced attack methods an actor can use, so it w be easy to conclude that attackers must be using special tricks and attack surfaces. But instead, the 0-days we saw in 2021 generally followed the same bug patterns, attack surfaces, and exploit “shapes” previously seen in public research. Once “0-day is hard”, we’d expect that to be successful, attackers would have to find new bug classes of vulnerabilities in new attack surfaces using never before seen exploitation methods. In general, that wasn't what the data showed us this year. With two exceptions (described below in the iOS section) out of the 58, everything we saw was pretty “meh” or standard.

Out of the 58 in-the-wild 0-days for the year, 39, or 67% were memory corruption vulnerabilities. Memory corruption vulnerabilities have been the standard for attacking software for the last few decades and it’s still how attackers are having success. Out of these memory corruption vulnerabilities, the majority also stuck with very popular and well-known bug classes:

- 17 use-after-free
- 6 out-of-bounds read & write
- 4 buffer overflow
- 4 integer overflow

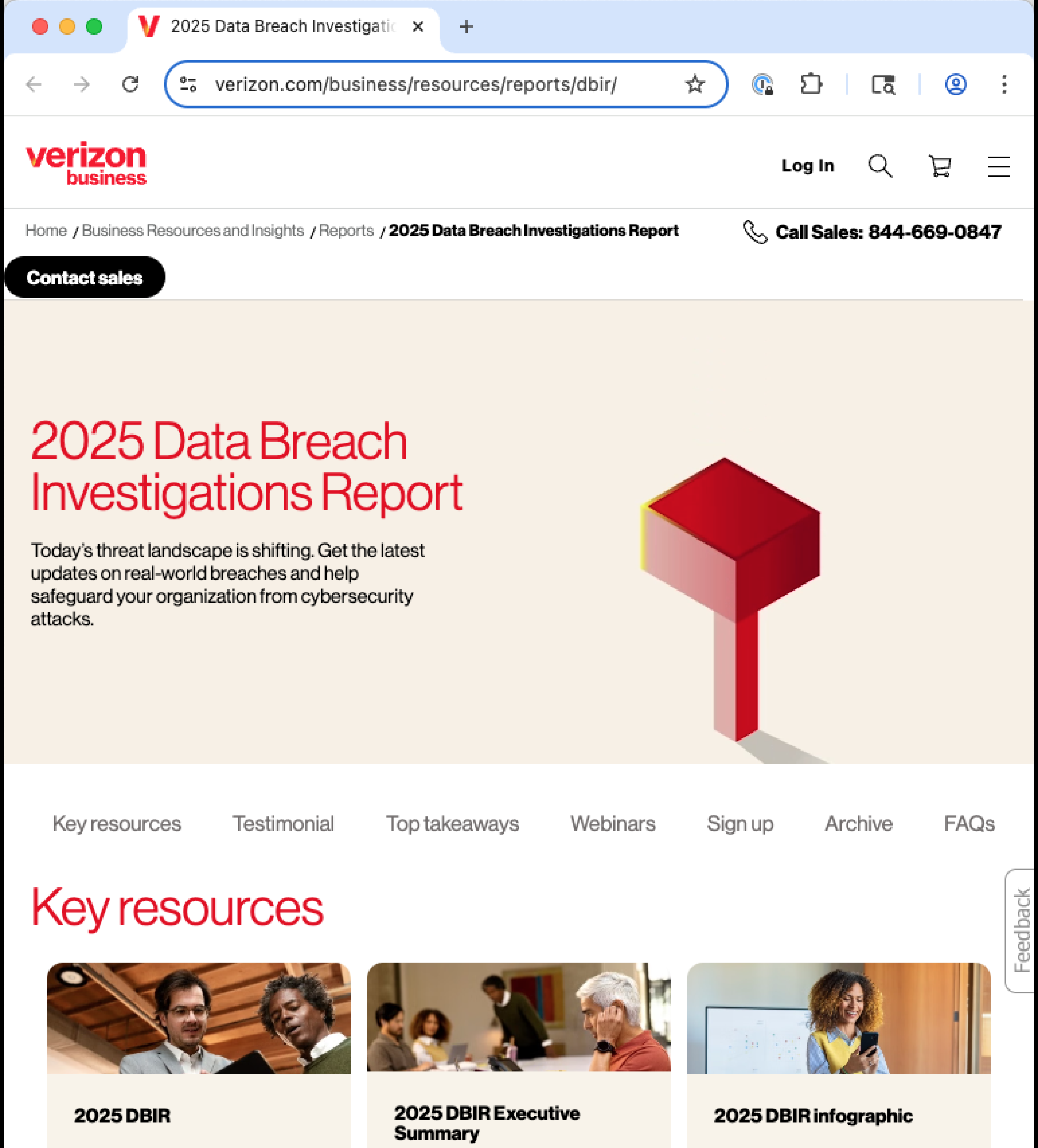


Out of the 58 in-the-wild 0-days for the year, 39, or 67% were memory corruption vulnerabilities.

How’s that going?

Lots we don’t know

- No estimate of the *damage* caused by the exploitation
- Data is *lower bound* – probably more exploited zero-days than these (not known, not reported)
- Zero-days are *not the only vector of attack*


2025 Data Breach Investigati
verizon.com/business/resources/reports/dbir/
verizon business
Log In
Home / Business Resources and Insights / Reports / 2025 Data Breach Investigations Report
Call Sales: 844-669-0847
Contact sales
Key resources
Testimonial
Top takeaways
Webinars
Sign up
Archive
FAQs
Feedback

2025 Data Breach Investigations Report

Today's threat landscape is shifting. Get the latest updates on real-world breaches and help safeguard your organization from cybersecurity attacks.

Key resources

2025 DBIR

2025 DBIR Executive Summary

2025 DBIR infographic

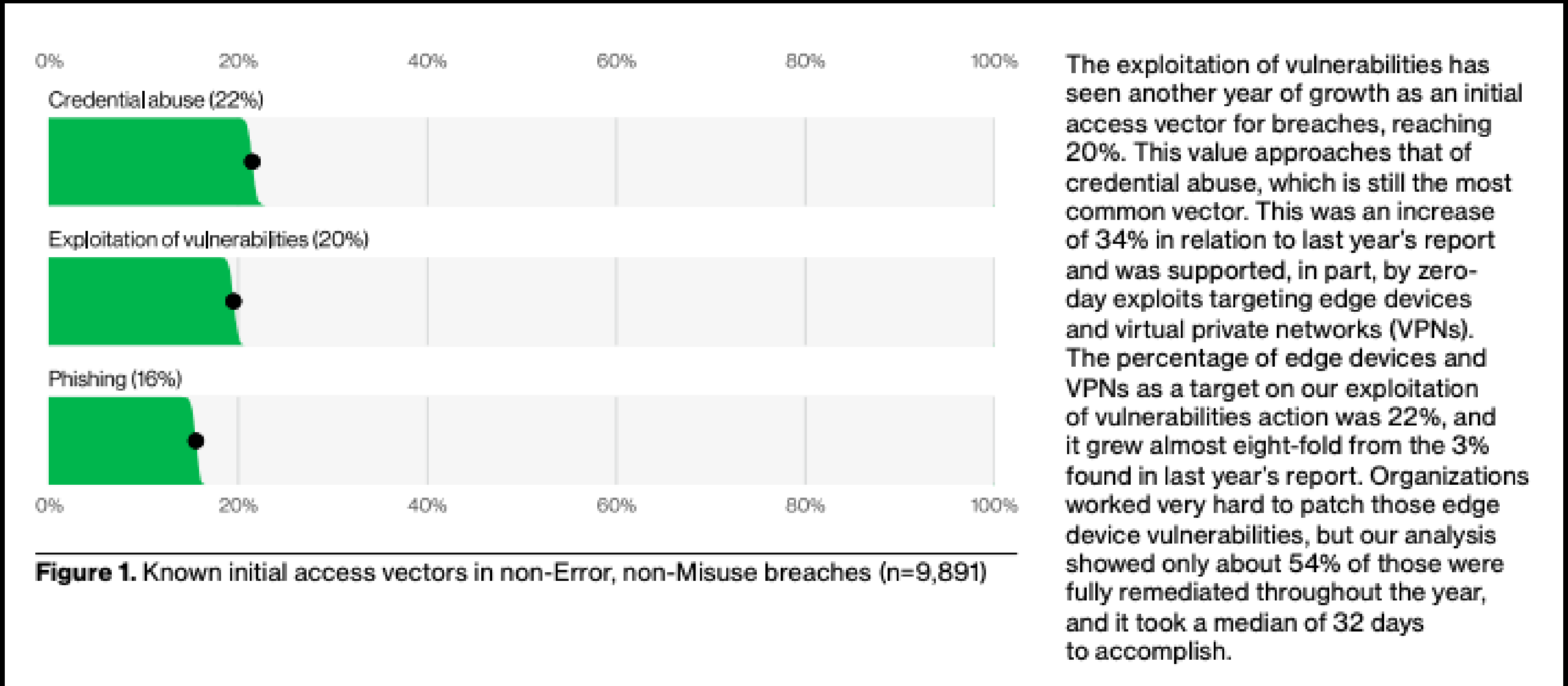
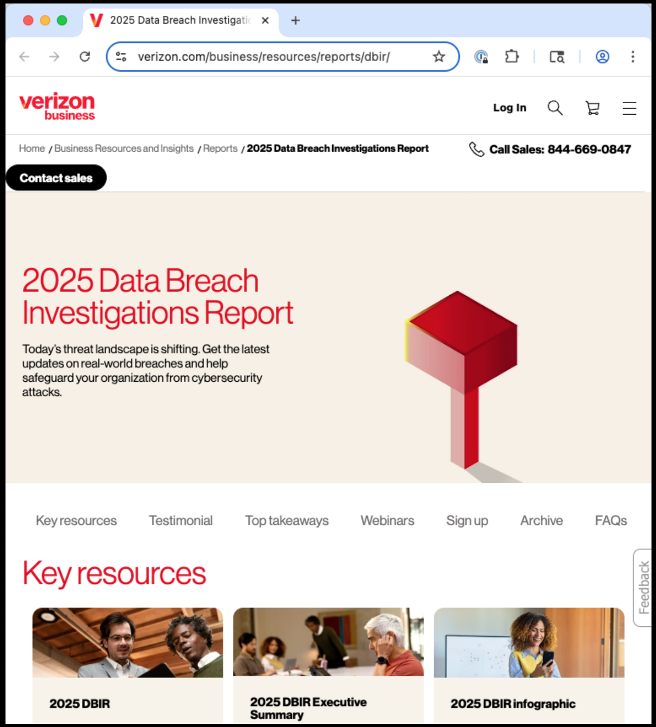


Figure 1. Known initial access vectors in non-Error, non-Misuse breaches (n=9,891)

The exploitation of vulnerabilities has seen another year of growth as an initial access vector for breaches, reaching 20%. This value approaches that of credential abuse, which is still the most common vector. This was an increase of 34% in relation to last year's report and was supported, in part, by zero-day exploits targeting edge devices and virtual private networks (VPNs). The percentage of edge devices and VPNs as a target on our exploitation of vulnerabilities action was 22%, and it grew almost eight-fold from the 3% found in last year's report. Organizations worked very hard to patch those edge device vulnerabilities, but our analysis showed only about 54% of those were fully remediated throughout the year, and it took a median of 32 days to accomplish.


2025 Data Breach Investigati
verizon.com/business/resources/reports/dbir/
verizon business
Log In
Home / Business Resources and Insights / Reports / 2025 Data Breach Investigations Report
Call Sales: 844-669-0847
Contact sales
Key resources
Testimonial
Top takeaways
Webinars
Sign up
Archive
FAQs
Feedback

2025 Data Breach Investigations Report

Today's threat landscape is shifting. Get the latest updates on real-world breaches and help safeguard your organization from cybersecurity attacks.

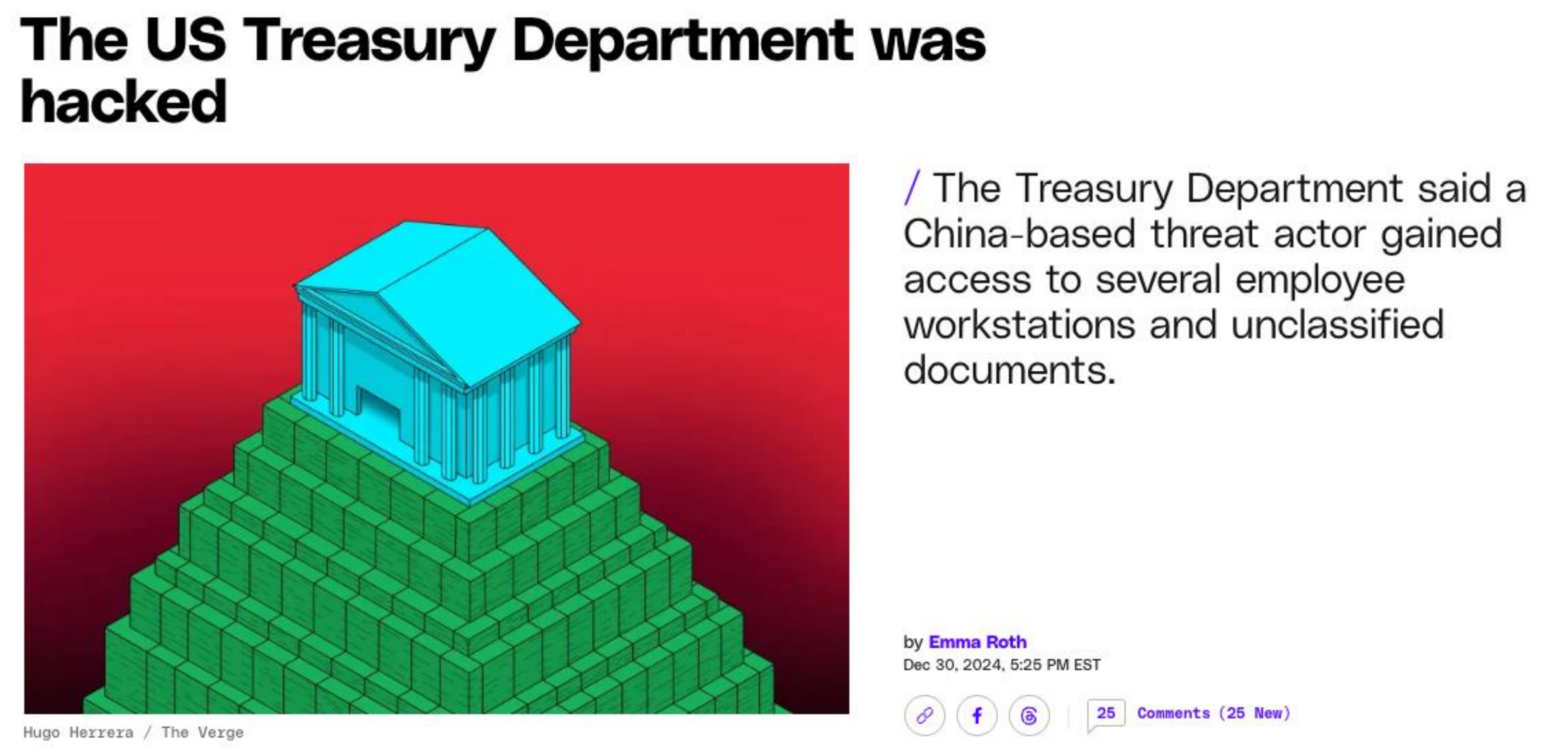
Key resources

2025 DBIR

2025 DBIR Executive Summary

2025 DBIR infographic

The exploitation of vulnerabilities has seen another year of growth as an initial access vector for breaches, reaching 20%. This value approaches that of credential abuse, which is still the most common vector.

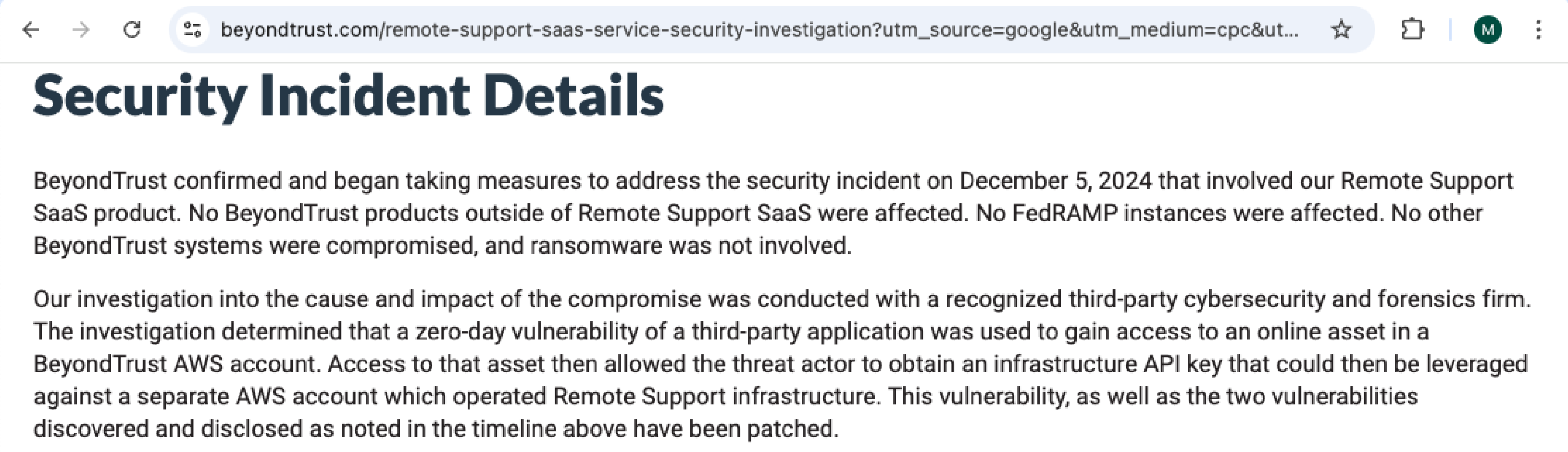


The US Treasury Department was hacked

/ The Treasury Department said a China-based threat actor gained access to several employee workstations and unclassified documents.

by **Emma Roth**
Dec 30, 2024, 5:25 PM EST

Hugo Herrera / The Verge



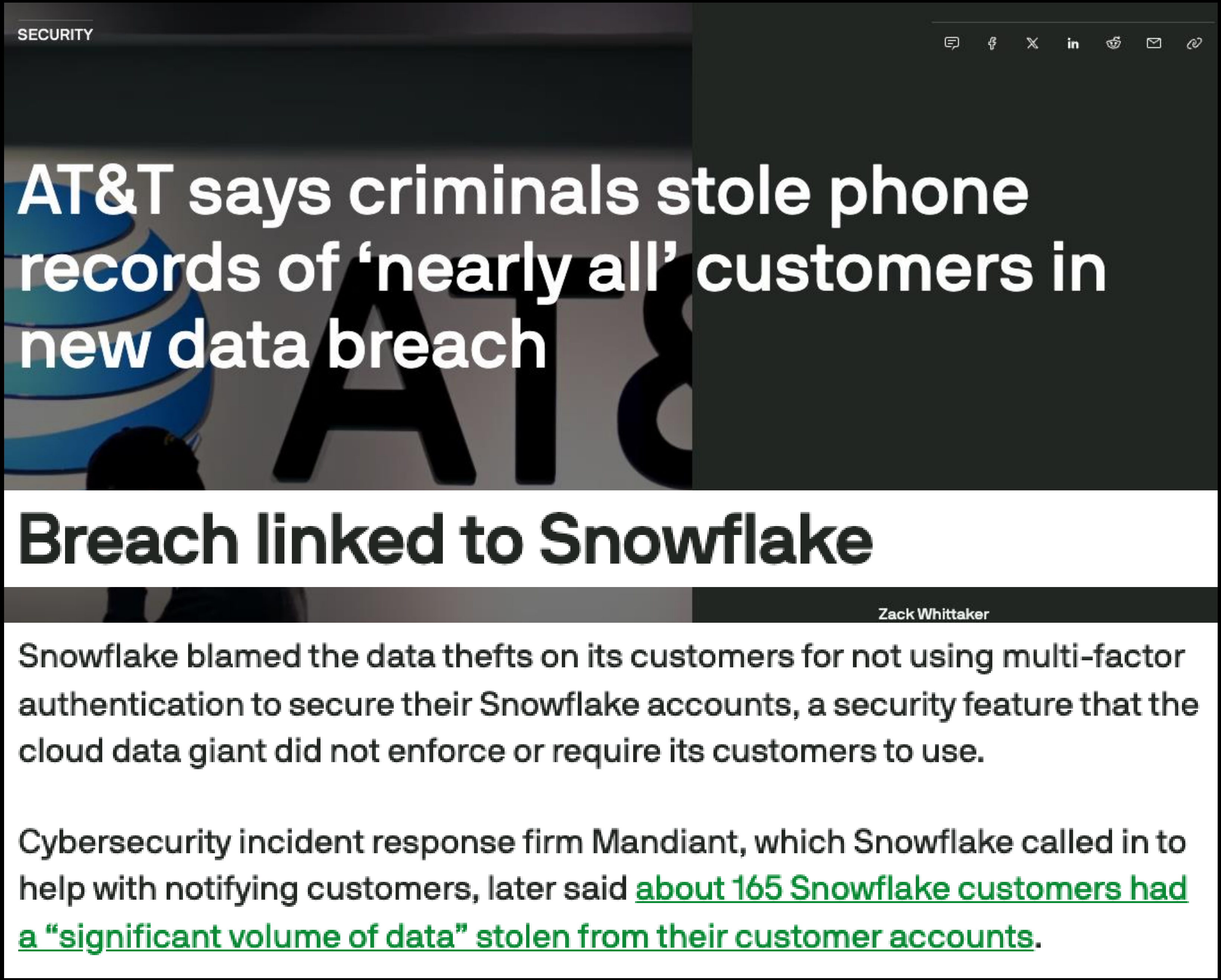
beyondtrust.com/remote-support-saas-service-security-investigation?utm_source=google&utm_medium=cpc&ut...
M

Security Incident Details

BeyondTrust confirmed and began taking measures to address the security incident on December 5, 2024 that involved our Remote Support SaaS product. No BeyondTrust products outside of Remote Support SaaS were affected. No FedRAMP instances were affected. No other BeyondTrust systems were compromised, and ransomware was not involved.

Our investigation into the cause and impact of the compromise was conducted with a recognized third-party cybersecurity and forensics firm. The investigation determined that a zero-day vulnerability of a third-party application was used to gain access to an online asset in a BeyondTrust AWS account. Access to that asset then allowed the threat actor to obtain an infrastructure API key that could then be leveraged against a separate AWS account which operated Remote Support infrastructure. This vulnerability, as well as the two vulnerabilities discovered and disclosed as noted in the timeline above have been patched.

… "gain access to an online asset in a BeyondTrust AWS account. Access to that asset then allowed the threat actor to **obtain an infrastructure API key** …" which was used to operate the Remote Support infrastructure



SECURITY

AT&T says criminals stole phone records of 'nearly all' customers in new data breach

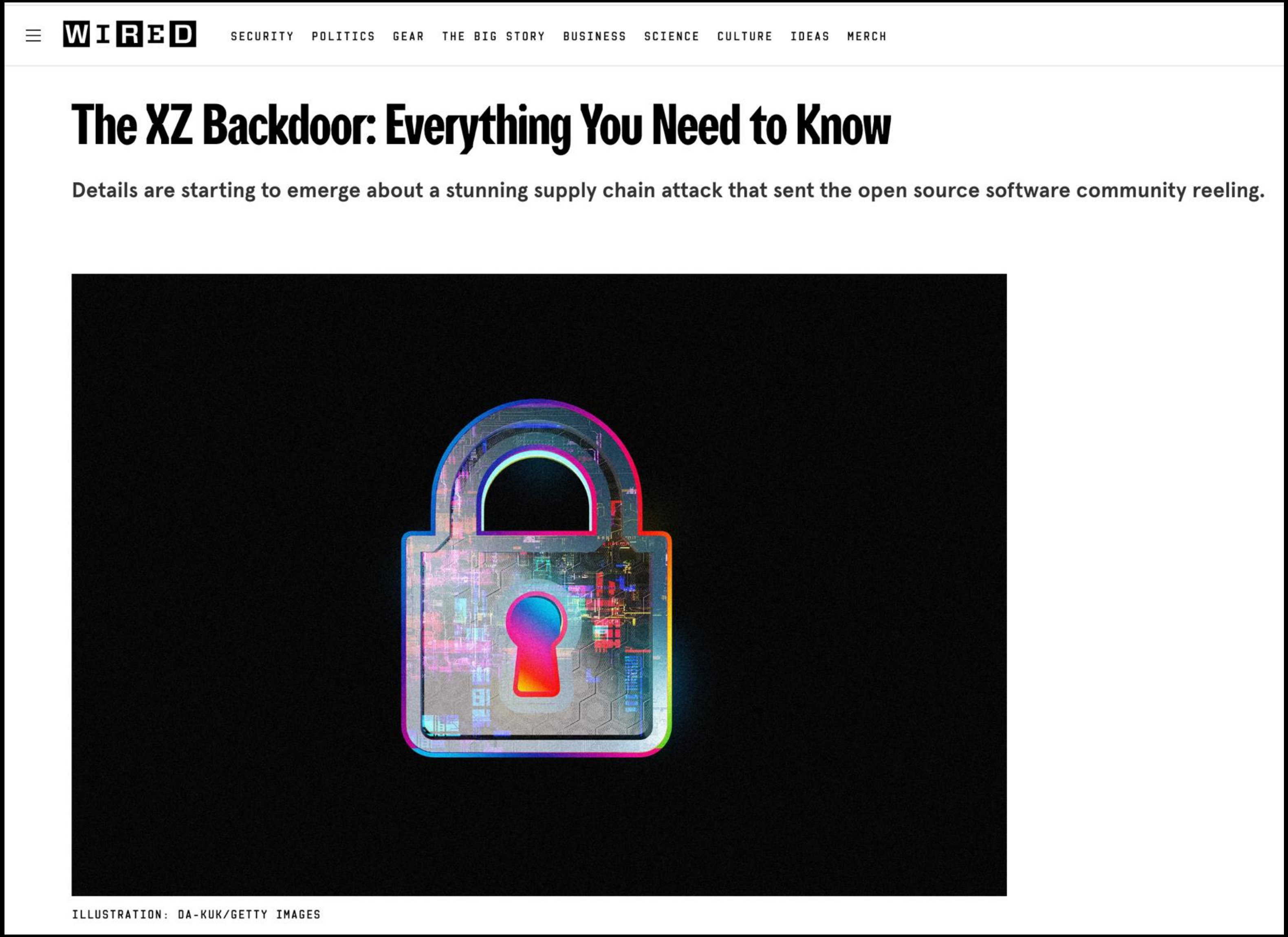
Breach linked to Snowflake

Zack Whittaker

Snowflake blamed the data thefts on its customers for not using multi-factor authentication to secure their Snowflake accounts, a security feature that the cloud data giant did not enforce or require its customers to use.

Cybersecurity incident response firm Mandiant, which Snowflake called in to help with notifying customers, later said about 165 Snowflake customers had a "significant volume of data" stolen from their customer accounts.

"Snowflake blamed the data thefts on its customers for **not using multi-factor authentication** to secure their snowflake accounts, … **did not require its customers to use**"



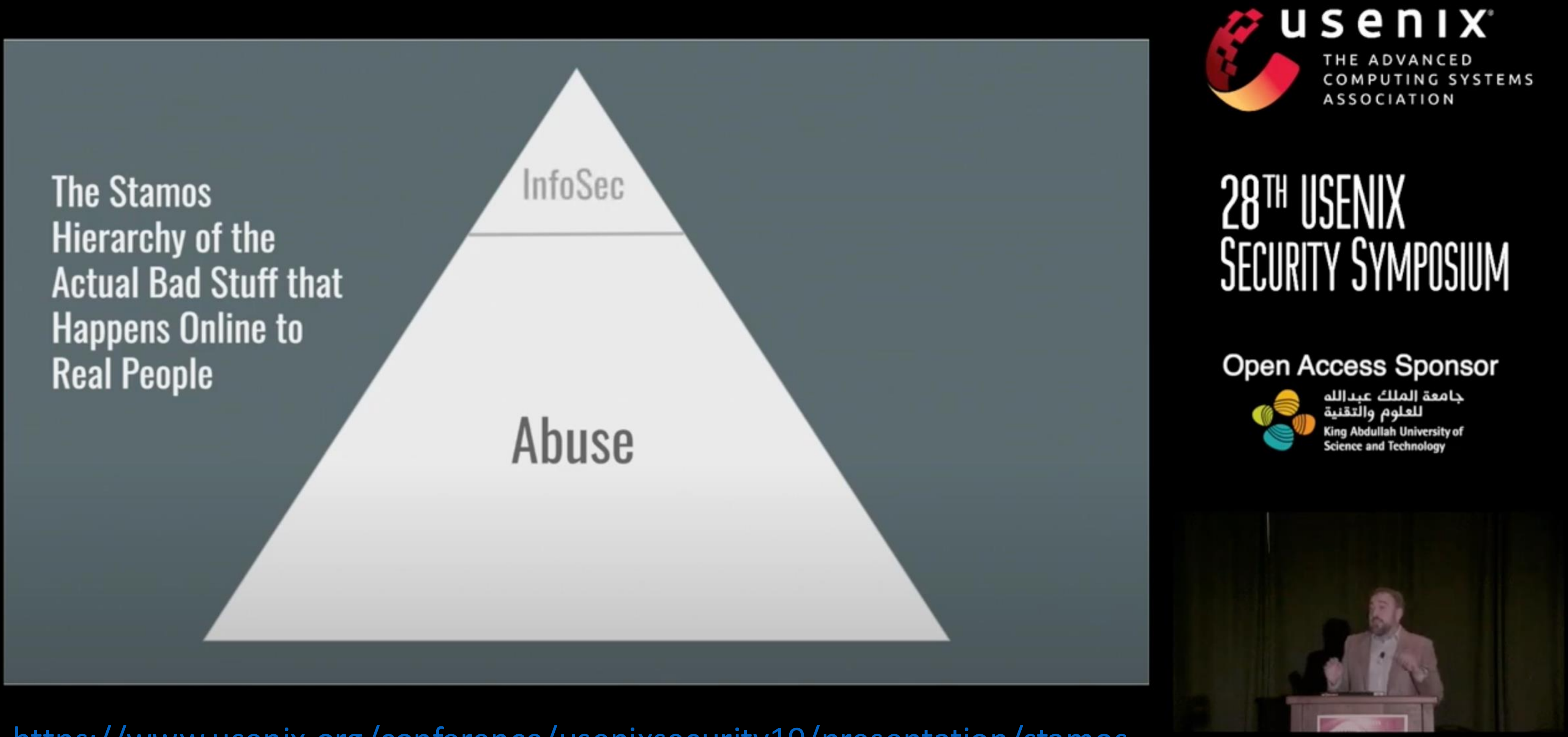
The XZ Backdoor: Everything You Need to Know

Details are starting to emerge about a stunning supply chain attack that sent the open source software community reeling.

ILLUSTRATION: DA-KUK/GETTY IMAGES

“This might be the best executed **supply chain attack** we've seen described in the open, and it's a nightmare scenario: **malicious, competent, authorized upstream in a widely used library**"

The big picture



The Stamos Hierarchy of the Actual Bad Stuff that Happens Online to Real People

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

28TH USENIX SECURITY SYMPOSIUM

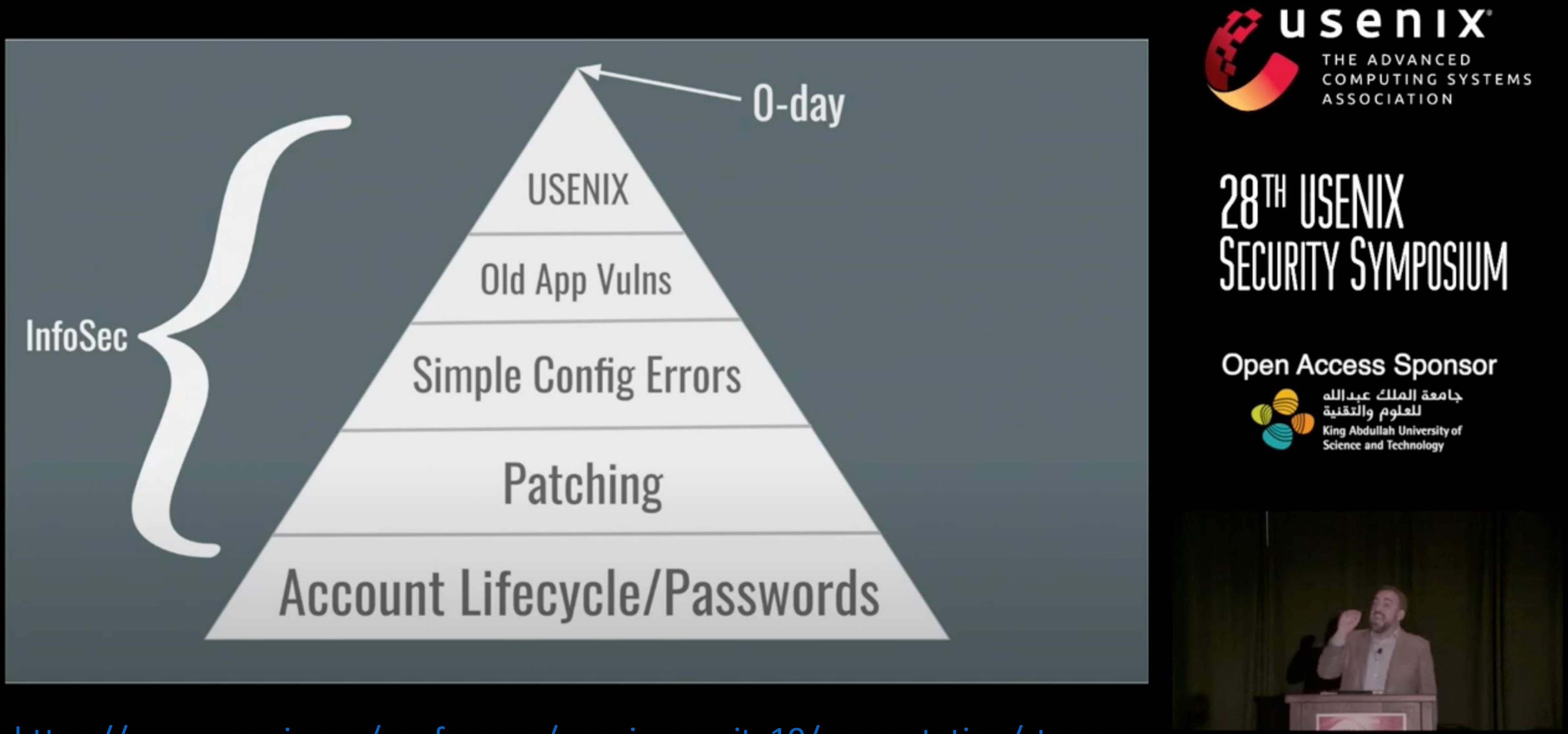
Open Access Sponsor

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

https://www.usenix.org/conference/usenixsecurity19/presentation/stamos

The big picture


0-day
USENIX
Old App Vulns
Simple Config Errors
Patching
Account Lifecycle/Passwords
InfoSec
usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION
28TH USENIX SECURITY SYMPOSIUM
Open Access Sponsor
جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

https://www.usenix.org/conference/usenixsecurity19/presentation/stamos

So: What would you do?

- If you were a CISO, or a VP of Security Engineering, how would you spend your money?
- If you were the head of a government agency like CISA, tasked with improving the state of cybersecurity, what would you recommend?
- If you were the head of NSF’s Secure and Trustworthy Cyberspace program, what would you fund?

Empirical Security & Privacy, for Humans

UPenn CIS 7000-010

Goals for the course

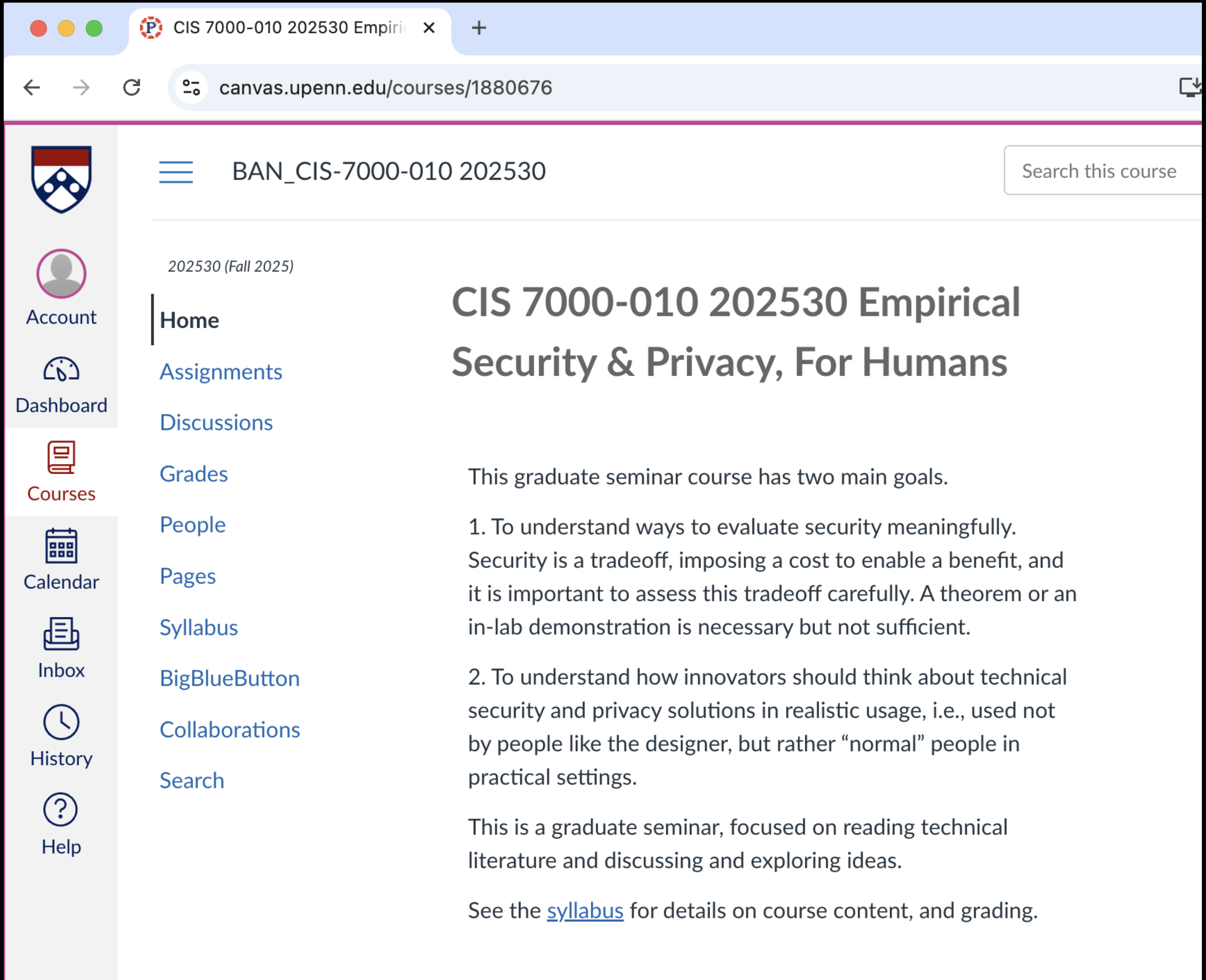
- Learn the research state of the art

- Economic view of cybersecurity
- End users and cybersecurity
- Cybersecurity as a scientific pursuit
- Cybersecurity investment as risk assessment
- Cybersecurity game theory
- Cyberattack economics
- Cybersecurity public health
- Developers' and operators' actions, and security
- Ethics in computer security experimentation
- Network-based security measurement
- Privacy

Goals for the course

- Learn the research state of the art
- Learn relevant research methods
- Learn how to learn – how to dive into a field, understand its results, and see gaps and opportunities
- Do something interesting: New result, reproduction, or a deep dive

https://canvas.upenn.edu/courses/1880676



CIS 7000-010 202530 Empirical Security & Privacy, For Humans

This graduate seminar course has two main goals.

1. To understand ways to evaluate security meaningfully. Security is a tradeoff, imposing a cost to enable a benefit, and it is important to assess this tradeoff carefully. A theorem or an in-lab demonstration is necessary but not sufficient.

2. To understand how innovators should think about technical security and privacy solutions in realistic usage, i.e., used not by people like the designer, but rather "normal" people in practical settings.

This is a graduate seminar, focused on reading technical literature and discussing and exploring ideas.

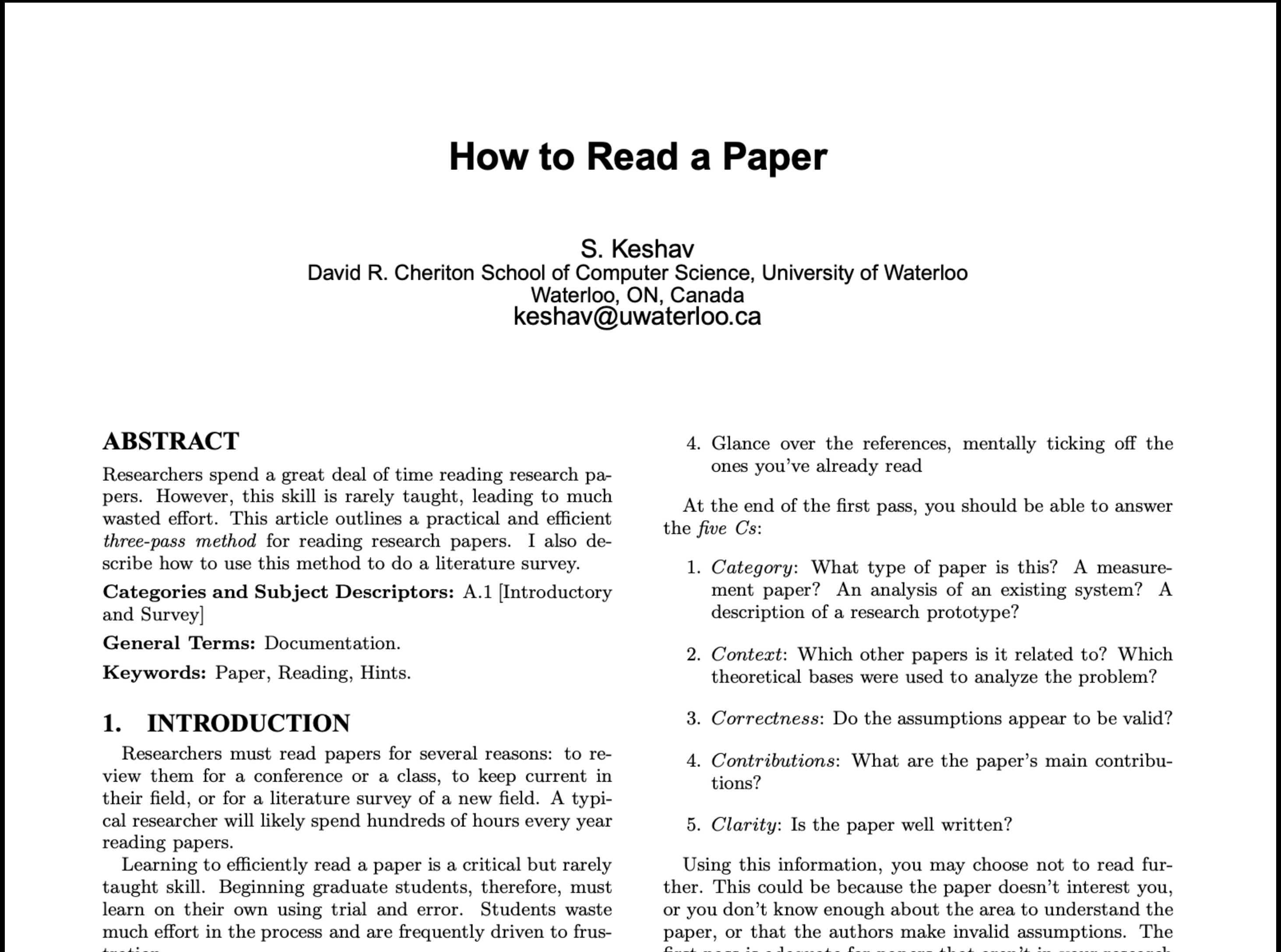
See the syllabus for details on course content, and grading.

Approach

- Read papers, (sometimes) present them, critique them, discuss
- Learn from experts in the field
- Do a (group) project

- We will have Zoom enabled during the class, for remote lectures and for those who (occasionally) can’t make it

Read papers: Question, understand, improve



How to Read a Paper

S. Keshav
David R. Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, Canada
keshav@uwaterloo.ca

ABSTRACT

Researchers spend a great deal of time reading research papers. However, this skill is rarely taught, leading to much wasted effort. This article outlines a practical and efficient *three-pass method* for reading research papers. I also describe how to use this method to do a literature survey.

Categories and Subject Descriptors: A.1 [Introductory and Survey]

General Terms: Documentation.

Keywords: Paper, Reading, Hints.

1. INTRODUCTION

Researchers must read papers for several reasons: to review them for a conference or a class, to keep current in their field, or for a literature survey of a new field. A typical researcher will likely spend hundreds of hours every year reading papers.

Learning to efficiently read a paper is a critical but rarely taught skill. Beginning graduate students, therefore, must learn on their own using trial and error. Students waste much effort in the process and are frequently driven to frustration.

4. Glance over the references, mentally ticking off the ones you've already read

At the end of the first pass, you should be able to answer the *five Cs*:

1. *Category*: What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?

2. *Context*: Which other papers is it related to? Which theoretical bases were used to analyze the problem?

3. *Correctness*: Do the assumptions appear to be valid?

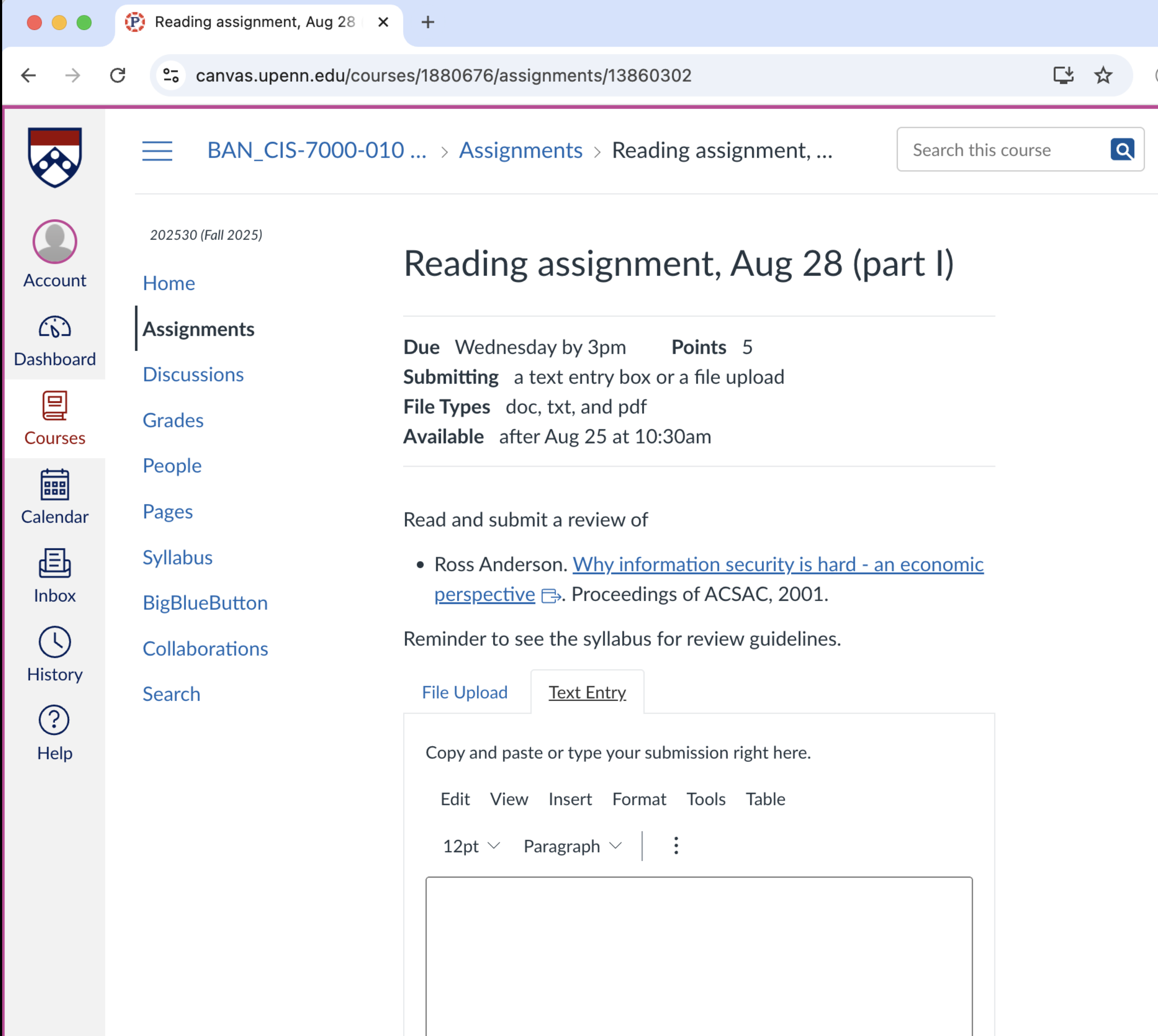
4. *Contributions*: What are the paper's main contributions?

5. *Clarity*: Is the paper well written?

Using this information, you may choose not to read further. This could be because the paper doesn't interest you, or you don't know enough about the area to understand the paper, or that the authors make invalid assumptions. The first pass is adequate for papers that aren't in your research

Class prep

- Read the paper(s) for that class. Submit a 1-2 paragraph review



Reading assignment, Aug 28 (part I)

Due Wednesday by 3pm **Points** 5
Submitting a text entry box or a file upload
File Types doc, txt, and pdf
Available after Aug 25 at 10:30am

Read and submit a review of

- Ross Anderson. Why information security is hard - an economic perspective. Proceedings of ACSAC, 2001.

Reminder to see the syllabus for review guidelines.

Copy and paste or type your submission right here.

Guest lectures (so far)

Alex Gantman
VP, Security Engineering,
Qualcomm, August 28

Cormac Herley
VP, Security Engineering,
Qualcomm, September 4

Adam Shostack
Founder & CEO, Shostack
& Associates, October 7

Present papers: Distill, reveal, dive deep

- Will do this for the second half of the class
- We will vote on a pool of papers to present, and you can select the 1 you want
- Grading criteria: Understanding, thoughtfulness, background/perspective, clarity, materials quality, delivery, non-regurgitation, answering questions

Projects

- Something substantial: New study, reproduction, literature review, …
- Timeline
  - Pitches in class @ 9/25
  - Proposal @ 10/9
  - Final paper @ finals week

About me

- Ph.D., CIS @ UPenn 2001
- Remained a Philly sports fan (go Eagles!)

About me

Employment

- 2002-2022 – TTk faculty, UMD
- 2006-2015 – Adjunct, IDA/CCS (NSA-funded research lab)
- 2008, 2015 – Visiting Researcher, Microsoft Research
- 2018-2021 – CTO, Correct Computation, Inc (startup)
- 2022-p
Scienti

adership positions

2 – Member, DARPA
on Science and
y Board

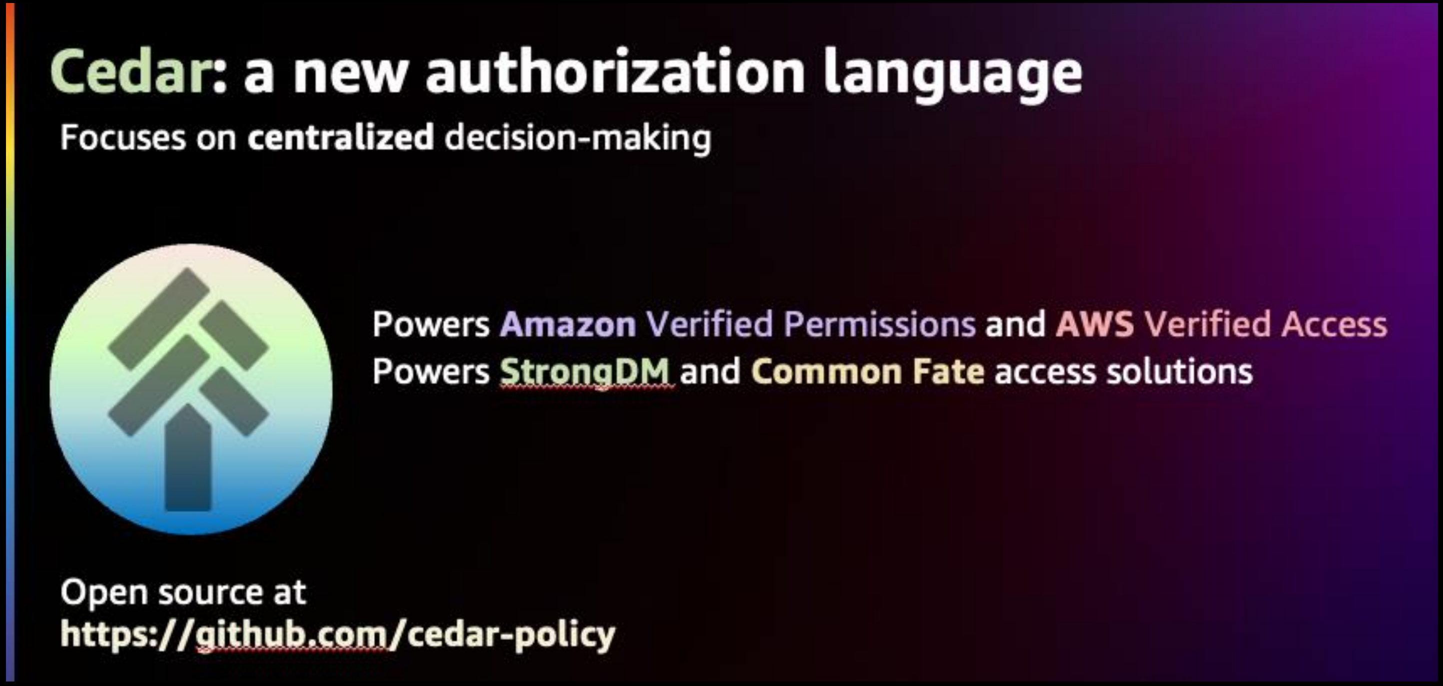
ctor, Maryland
nter

ir, ACM Special
roup on Programming
s

L
,
d Education, UMD CS

About me

- Research @ UMD: **Software Security**, Programming Languages, Software Engineering, Usability, Cryptography, Quantum Computing, Networks, Databases
- Startup: Building tools for secure software development
  - Binary analysis
  - Migration to memory-safe C
- AWS
  - Cedar authorization language
  - Fuzzing/automated test generation
  - Formal/mechanized proofs of security



Cedar: a new authorization language

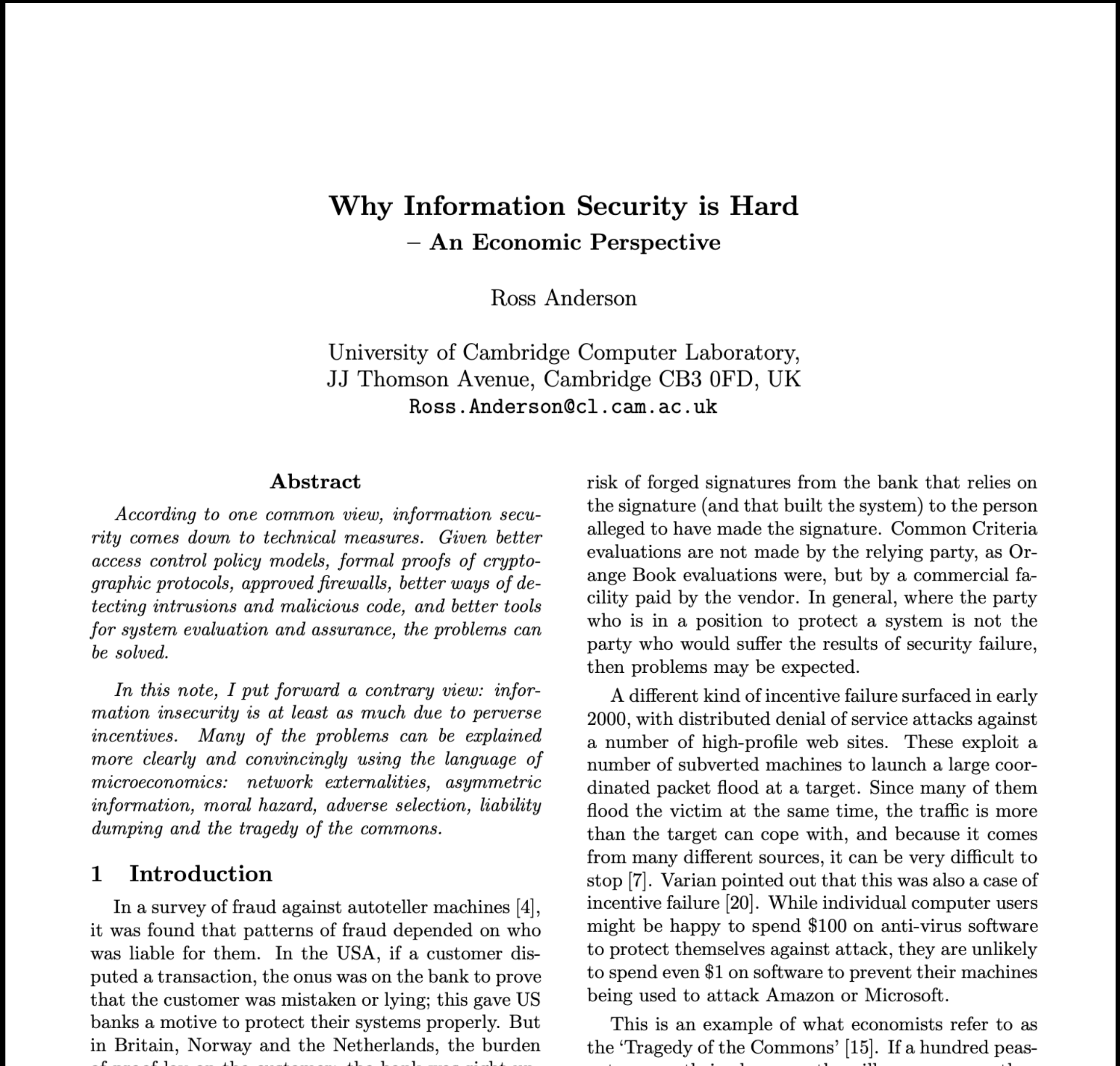
Focuses on **centralized** decision-making

Powers **Amazon** Verified Permissions and **AWS** Verified Access
Powers **StrongDM** and **Common Fate** access solutions

Open source at
https://github.com/cedar-policy

Reading for next time

Plus: “How to Read a Paper?”



Why Information Security is Hard – An Economic Perspective

Ross Anderson

University of Cambridge Computer Laboratory,
JJ Thomson Avenue, Cambridge CB3 0FD, UK
Ross.Anderson@cl.cam.ac.uk

Abstract

According to one common view, information security comes down to technical measures. Given better access control policy models, formal proofs of cryptographic protocols, approved firewalls, better ways of detecting intrusions and malicious code, and better tools for system evaluation and assurance, the problems can be solved.

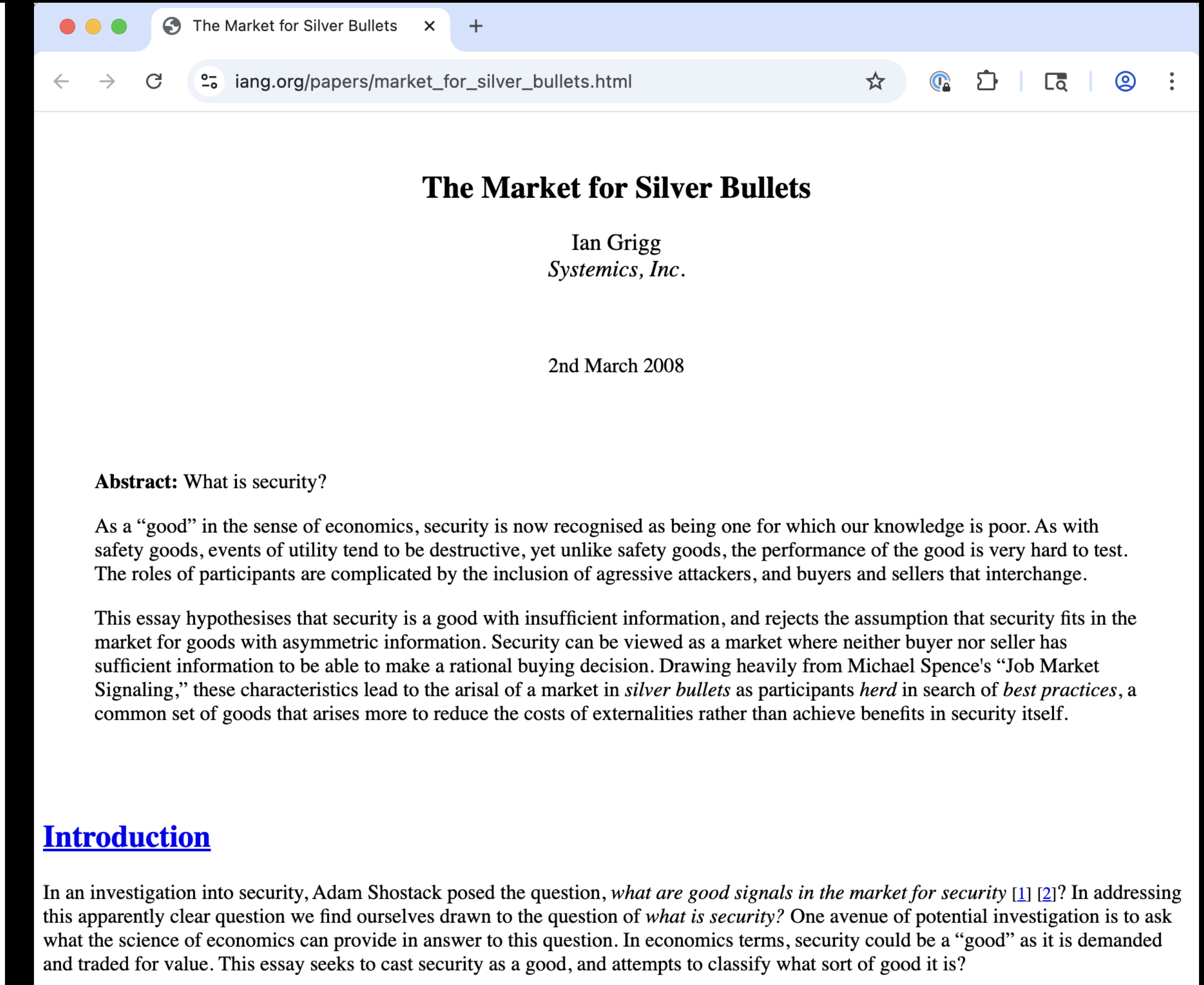
In this note, I put forward a contrary view: information insecurity is at least as much due to perverse incentives. Many of the problems can be explained more clearly and convincingly using the language of microeconomics: network externalities, asymmetric information, moral hazard, adverse selection, liability dumping and the tragedy of the commons.

1 Introduction

In a survey of fraud against autoteller machines [4], it was found that patterns of fraud depended on who was liable for them. In the USA, if a customer disputed a transaction, the onus was on the bank to prove that the customer was mistaken or lying; this gave US banks a motive to protect their systems properly. But in Britain, Norway and the Netherlands, the burden risk of forged signatures from the bank that relies on the signature (and that built the system) to the person alleged to have made the signature. Common Criteria evaluations are not made by the relying party, as Orange Book evaluations were, but by a commercial facility paid by the vendor. In general, where the party who is in a position to protect a system is not the party who would suffer the results of security failure, then problems may be expected.

A different kind of incentive failure surfaced in early 2000, with distributed denial of service attacks against a number of high-profile web sites. These exploit a number of subverted machines to launch a large coordinated packet flood at a target. Since many of them flood the victim at the same time, the traffic is more than the target can cope with, and because it comes from many different sources, it can be very difficult to stop [7]. Varian pointed out that this was also a case of incentive failure [20]. While individual computer users might be happy to spend $100 on anti-virus software to protect themselves against attack, they are unlikely to spend even $1 on software to prevent their machines being used to attack Amazon or Microsoft.

This is an example of what economists refer to as the ‘Tragedy of the Commons’ [15]. If a hundred peas-



The Market for Silver Bullets
×
+
iang.org/papers/market_for_silver_bullets.html

The Market for Silver Bullets

Ian Grigg
Systemics, Inc.

2nd March 2008

Abstract: What is security?

As a “good” in the sense of economics, security is now recognised as being one for which our knowledge is poor. As with safety goods, events of utility tend to be destructive, yet unlike safety goods, the performance of the good is very hard to test. The roles of participants are complicated by the inclusion of agressive attackers, and buyers and sellers that interchange.

This essay hypothesises that security is a good with insufficient information, and rejects the assumption that security fits in the market for goods with asymmetric information. Security can be viewed as a market where neither buyer nor seller has sufficient information to be able to make a rational buying decision. Drawing heavily from Michael Spence's “Job Market Signaling,” these characteristics lead to the arisal of a market in *silver bullets* as participants *herd* in search of *best practices*, a common set of goods that arises more to reduce the costs of externalities rather than achieve benefits in security itself.

Introduction

In an investigation into security, Adam Shostack posed the question, *what are good signals in the market for security* [1] [2]? In addressing this apparently clear question we find ourselves drawn to the question of *what is security?* One avenue of potential investigation is to ask what the science of economics can provide in answer to this question. In economics terms, security could be a “good” as it is demanded and traded for value. This essay seeks to cast security as a good, and attempts to classify what sort of good it is?