Quantifying Memory Unsafty and Reactions to It

Alex Gaynor, Fish in a Barrel
Fish in a Barrel, not a real company
Someone has your password

Hi John

Someone just used your password to try to sign in to your Google Account johnpodesta@gmail.com.

Details:
Saturday, 19 March, 8:34:30 UTC
IP Address: 134.249.139.239
Location: Ukraine

Google stopped this sign-in attempt. You should change your password immediately.

CHANGE PASSWORD

John Podesta, 2016
Security keys
Account takeover prevention rates, by challenge type

Device-based challenges:
- On-device prompt: 100% automated bot, 99% bulk phishing attack, 90% targeted attack
- SMS code: 100% automated bot, 96% bulk phishing attack, 76% targeted attack
- Security key: 100% automated bot, 100% bulk phishing attack, 100% targeted attack

Knowledge-based challenges:
- Secondary email address: 73% automated bot, 68% bulk phishing attack, 79% targeted attack
- Phone number: 100% automated bot, 26% bulk phishing attack, 50% targeted attack
- Last sign-in location: 100% automated bot, 100% bulk phishing attack, 10% targeted attack

95% confidence interval
Memory Unsafety
Properties of memory unsafety

- **Spatial:**
  - Buffer overflow (heap or stack, read or write)

- **Temporal:**
  - Use-after-free
  - Use of uninitialized memory
  - Wild pointer dereference

- **Type confusion**
Languages

Memory safe:

- Rust
- Swift
- Python
- Java
- Go
- etc.

Memory unsafe:

- C
- C++
- Assembly
Case studies

- iOS 0-day (and n-day) exploits used against the Uighurs
- iOS and Android n-day exploits used against Tibetans
- iOS 0-day exploits used against Ahmed Mansoor
- WhatsApp 0-day exploit, with varied targets
- WannaCry
- HeartBleed
The stages of grief
Denial Symptoms:

“Programming in memory unsafe languages does not cause an increased rate of vulnerabilities.”
Denial: Data

- **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 unsafety
- **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
The vulnerability venn diagram

Safe languages

Unsafe languages
Anger symptoms:

“Yes, code in memory unsafe languages can have bugs. But if you were a better programmer, you wouldn’t have this problem.”
Anger: Complex systems

How Complex Systems Fail

(Being a Short Treatise on the Nature of Failure; How Failure is Evaluated; How Failure is Attributed to Proximate Cause; and the Resulting New Understanding of Patient Safety)

-- https://how.complexsystems.fail/
Bargaining symptoms:

“Ok, yes, memory unsafety is a problem. But surely we can address it with static analysis and fuzzing and sandboxing and mitigations and red-teaming.”
Bargaining: Response

- Chrome: Tens of thousands of fuzzing cores
- iOS: Every single app is sandboxed
- Windows: Extensive exploit mitigations, including KCFG
- Chrome: Aggressive multi-process sandboxed design
- All: Millions of dollars spent on bug bounties
Depression symptoms:

“Memory unsafety is a problem... but oh my god we have a trillion lines of C/C++, we can never rewrite all of it, everything is hopeless.”
Depression: Work smarter, not harder

- Identify high leverage places
  - Code that runs with high privileges
  - Code that acts as a key part of a security guarantee
  - Code that has a large user-accessible attack surface
Acceptance symptoms:
Asking how, not if.
A call to action

- Build a coalition who recognizes the gravity of this problem
- Find a memory safe language that’s a good fit for your domain
- Stop the bleeding: make it possible for new code bases in your organization to be memory safe
- Find your highest leverage attack surfaces in existing memory unsafe code and get to work!
- Use language as a factor when assessing the security of projects
Proof that incremental migrations are possible

- Python Cryptographic Authority
- Rust-For-Linux
- Firefox
- Librsvg

Your project can be next!
Fin
Questions?

https://alexgaynor.net
Citations and references

1. https://security.googleblog.com/2019/05/new-research-how-effective-is-basic.html
11. https://how.complexsystems.fail/