MiraclePtr One Pager

(go/miracleptr)

mailing list: chrome-memory-safety@google.com
contact: haraken@, bartekn@

2021 July

STATUS: PUBLIC

This is a one-pager to collect all documents and resources about MiraclePtr<T>. This document is editable for chrome-memory-safety@ -- feel free to add your documents :)

TL;DR of the project

Chrome's biggest security problem is a constant stream of exploitable (and exploited) Use-after-Frees. Our goal is to stop them being exploitable by **turning them from security bugs to non-security crashes or memory leaks**.

- MiraclePtr rewrites member field raw pointers in the Chromium repository (and eventually some third party libraries). For performance reasons, function parameters, on-stack variables etc are not the target of the rewrite. **The rewrite is tool-assisted** and 99% is done by the auto rewriter. The rewrite can be rolled out incrementally on a per-directory basis. [Note: The rewrite is not needed for PCScan / DCScan.]
- The raw pointer => MiraclePtr rewrite is mostly orthogonal to the underlying implementation of MiraclePtr. We can consider enabling different underlying implementations on different processes and platforms.
- MiraclePtr can be used for other purposes like pointer compression.
- MiraclePtr needs memory allocator support and **depends on PartitionAlloc-Everywhere** (i.e., route malloc() in Chrome to PartitionAlloc). PartitionAlloc-Everywhere has its own benefits: fewer allocators in Chrome and improved security.
- ARM has a plan to support hardware MTE and Android has a plan to support it from "S". We are discussing with ARM to support it in PartitionAlloc. Once it's supported, (some of) our software implementation will move to hardware. The software implementation is still needed for non-ARM CPU devices / desktops.
- The launch criteria is: no significant regression in performance, memory, binary size and stability. We will not launch MiraclePtr until we meet the launch criteria.
- The primary goal is to stop UaF bugs being exploitable. The diagnosability (i.e., getting actionable stack traces) is the secondary goal. For that purpose, we have a better tool like GWP-ASan. MiraclePtr is the backstop to ensure that UAF crashes are not exploitable.

- Per the feedback from Project Zero, we are exploring two variants: **BackupRefPtr (deterministic) and PCScan (mostly deterministic) / DCScan (deterministic)**. We are planning to send the two variants to the real-world A/B experiments and collect numbers.
- We want to **launch things incrementally**. We start with **the browser process on Windows 64 bit** because 1) the browser process is less performance-sensitive and more security-sensitive, 2) Windows supports a binary A/B experiment and 3) 64 bit is easier to support than 32 bit and is where the majority of users lives. Then we iterate to support renderer processes and more platforms. However, **it's important to have a plan to support renderer processes and all platforms including 32 bit** because otherwise we cannot lower the severity of a given UaF bug, which is key to reduce Stable respins and the workload on the Chrome team.

For more details, see the documents below.

Documents

Presentations & articles

- OWND 0x001B article
- lukasza@'s and bartekn@'s MiraclePtr breakout talk BlinkOn 13
  - Slides
  - Video recording
- lukasza@'s MiraclePtr presentation for Chrome-WA (links are Google-internal, sorry):
  - Slides
  - Video recording (MiraclePtr starts around 16:15 timestamp)
- bartekn@'s MiraclePtr presentation for Q3 ENG JP All Hands (links are Google-internal, sorry):
  - Slides
  - Video recording (MiraclePtr starts around 36:38 timestamp)
- adetaylor@'s and lukasza@'s infodump presentation for chrome-security@ (links are Google-internal, sorry):
  - Slides
- bartekn@'s MiraclePtr lightning talk for BlinkOn 15
  - Slides
  - Video recording (MiraclePtr starts around 31:17, it's preceded by an interesting talk about PartitionAlloc-Everywhere)
- keishi@'s and bartekn@'s breakout talk at BlinkOn 16
  - Slides
  - Video recording

Pointer safety proposals

- Protection for pointers on heap:
  - Pointer safety ideas - comparison of proposals (bartekn@, lukasza@, haraken@)
- Protection for pointers on the stack:
  - Deterministic stack scanning (haraken@)
  - BorrowRefPtr (markbrand@; this is an extension of BackupRefPtr)

Important documents

- MiraclePtr Security Impact [Googlers only, sorry, but summary here] (adetaylor@, lukasza@, palmer@, tsepez@)
- MiraclePtr Security Properties Comparison [adetaylor@, markbrand@]
- Eng Review:
  - MiraclePtr / Scan Eng Review one pager (2021 Aug) (haraken@)
  - MiraclePtr Roadmap (haraken@, bartekn@, keishi@)
  - Scan Roadmap (hpayer@, mlippautz@)
- raw_ptr.md - usage rules (bartekn@, lukasza@)
- BackupRefPtr Support Coverage (bartekn@ et al)
- BackupRefPtr 2022 Roadmap (keishi@, bartekn@)

Other / old documents

- Team meeting notes
- PartitionAlloc Everywhere (lizeb@)
- BigRewrite-related docs:
  - rewrite_raw_ptr_fields clang tool (lukasza@, bartekn@)
  - rewrite instructions (incl. post-rewrite steps) (bartekn@, keishi@)
  - Applying MiraclePtr (API draft & transition challenges) (bartekn@, lukasza@)
  - Comparison of rewrite choices - one-time BigRewrite VS on-the-fly/build-time rewrite VS LLVM integration (lukasza@)
  - Big Rewrite vs. LLVM Rewrite (lukasza@)
  - Big Rewrite execution: PSA, checklist, issues, tracking bug (keishi@ et al)
- Security-related docs
  - Initial Feedback from Project Zero [Googlers only, sorry] (markbrand@, glazunov@)
  - MiraclePtr Security Impact - see the link in “Important documents” section above
  - M78-M80 UaF in security bugs [Googlers only, sorry - I’d love to share more broadly, but some of the security bugs are not yet public]
  - M82-M90 UaF [Googlers only, sorry]
  - 100 most recent security bugs [Googlers only, sorry] (adetaylor@)
  - [somewhat old/stale] Sheriffing algorithm proposal (markbrand@)

- CheckedPtr/BackupRefPtr Limitations (bartekn@, lukasza@, glazunov@)
- Esoteric issues encountered when working with BRP (bartekn@ et al)
- raw_ptr clang plugin (keishi@)
- NoOpImpl overhead (yukiy@)
- crbug.com/1272324 - master bug to track the process of landing Big Rewrite
- Implementation ideas:
    - BackupRefPtr design
    - PartitionTag per PartitionAlloc object [Googlers only, sorry] (tasak@)
    - BackupRefPtr on top of ASan (markbrand@ and glazunov@)
    - Dangling pointer detector (haraken@)
    - MiraclePtr Perf Improvement Ideas (glazunov@, bartekn@, tasak@, lukasza@)
    - How to solve the address reuse problem (haraken@)
- Performance evaluation:
    - CheckedPtr2 Performance on micro-benchmarks (lizeb@)
    - PA Tag Perf Analysis (tasak@)
    - Initial performance / memory / binary size results (bartekn@)
    - Initial Renderer results on benchmarks (tasak@ keishi@)
    - … see more in the “Experiments” section
- Experiments:
    - Initial Experimentation Plan [Googlers only, sorry] (bartekn@)
    - BRP Canary "Pulse" experiments (keishi@, bartekn@)
    - BRP Binary A/B Dev experiments (keishi@, bartekn@, yukishiino@)
        - Experiment Release Timelines [Googlers only] (srinivassista@)
        - BRP experiment results (Dev) [Googlers only] (keishi@ et al)
    - BRP Binary A/B Stable & Beta experiments (bartekn@, keishi@)
        - Stable & Beta experiment plan [Googlers only, sorry] (bartekn@)
        - Stable & Beta experiment population setup [Googlers only] (bartekn@)
        - Stable & Beta Experiment Release Flow [Googlers only] (srinivassista@)
        - BRP experiment results for Beta [Googlers only] (bartekn@)
        - BRP experiment results for Stable [Googlers only] (bartekn@, keishi@, mpearson@)
    - New experiment model:
        - Design [Googlers only, sorry] (bartekn)
        - Results for BRP support compiled in, but disabled at run-time (Dev): Google-internal, public
        - Finch experiments [Googlers only, sorry] (bartekn)
- Testing plan for Eng Review (haraken@, bartekn@, lukasza@)
- MiraclePtr: Post Eng Review Roadmap (bartekn@)
- Initial Eng Review:
    - MiraclePtr decision tree (haraken@ et al)
    - Eng Review one pager (haraken@)
    - Eng Review deck (haraken@ et al)
    - Meeting notes

- Final Eng Review:
    - Eng review deck (haraken@ et al)
    - Pre-review Q&A
    - Meeting notes
    - See more in Important documents