V8 Sandbox - Glossary

Author: saelo@
First Published: December 2023
Last Updated: December 2023
Status: Living Doc
Visibility: **PUBLIC**

This document is part of the V8 Sandbox Project and explains the technical terms used in the context of the sandbox. A version of this document can also be found in the V8 source tree at src/sandbox/GLOSSARY.md. For a general overview of the sandbox design see the high-level design document.

General

Sandbox

Summary: A region of virtual address space (typically 1TB) containing all untrusted V8 objects.
Implementation: This is implemented as a large, contiguous virtual address space reservation using the appropriate operating system primitives. Address space reservations are cheap on all modern OSes.
Security Properties: It is assumed that an attacker can arbitrarily read and write memory inside the sandbox address space due to a typical security bug in V8. The mechanisms described in this document, in particular the various pointer types, then attempt to prevent an attacker from corrupting other memory inside the process running V8.
Design Document: V8 Sandbox - Address Space

Objects

This section gives an overview of the different object types related to the sandbox.

TrustedObject

Summary: A V8 HeapObject containing sensitive data or code, located outside of the sandbox.

Implementation: These are regular HeapObjects with a special instance type and which are allocated in one of the trusted heap spaces, located outside of the sandbox.
Security Properties: As these objects live outside of the sandbox, it can be assumed that they have not been manipulated by an attacker. As such, they can safely be read from, but one must be particularly careful when writing to them to avoid any memory corruption in trusted space.
Design Document: V8 Sandbox - Trusted Space

ExposedTrustedObject

Summary: A trusted object directly exposed to objects inside the sandbox.
Implementation: These objects own an entry in one of the trusted pointer tables and therefore can be referenced from inside the sandbox in a memory-safe way via a trusted pointer (see below).
Security Properties: Same as TrustedObject.
Design Document: V8 Sandbox - Trusted Space

Pointers

This section gives an overview of the different pointer types used by the sandbox.

Compressed Pointer

Summary: A pointer that is guaranteed to point into V8's 4GB heap area, inside the sandbox.
Implementation: These are stored as 32-bit offsets from the start of the heap area. They were originally developed to reduce V8's memory footprint.
Security Properties: As this pointer always points into the sandbox, it can always safely be written to, but when reading from it, it must be assumed that the data has been corrupted by an attacker.
Design Document: Compressed pointers in V8 (public doc)

Uncompressed/Full Pointer

Summary: A full (64-bit) pointer to a V8 HeapObject.

Implementation: These are regular (“raw”) pointers to HeapObjects. They appear for example after decompressing a compressed pointer when loading it into a register or onto the stack, or as fields in
objects located outside of the sandbox, for example tracking data structures used by the GC or the compilers, or objects created by the Embedder.
Security Properties: As these are raw pointers, they must not be corrupted by an attacker and so must only be used outside of the sandbox. As they also point into the sandbox, the same considerations as for Compressed Pointers apply, namely that it must be assumed that what they point to has been corrupted.

Sandboxed Pointer

Summary: A pointer that is guaranteed to point into the sandbox.
Implementation: A sandboxed pointer is a 40-bit offset (when the sandbox is 1TB large) that is added to the base of the sandbox when loaded. When the sandbox is disabled, they are regular full pointers.
Security Properties: As this pointer always points into the sandbox, it can always safely be written to, but when reading from it, it must be assumed that the data has been corrupted by an attacker.
Design Document: V8 Sandbox - Sandboxed Pointers

Protected Pointer

Summary: A pointer that cannot be modified by an attacker. Only used between TrustedObjects.
Implementation: These are implemented as compressed pointers but use the trusted space base instead of the in-sandbox heap base. It’s therefore guaranteed that they will always point into trusted space. As the pointer itself is also located in trusted space, it cannot directly be modified by an attacker. When the sandbox is disabled, they are regular tagged pointers.
Security Properties: Neither the pointer itself nor the pointed-to object can be modified by an attacker as both live in trusted space. These pointers therefore have the strongest security guarantees.
Design Document: V8 Sandbox - Trusted Space

External Pointer

Summary: A pointer to an object external to V8, outside of the sandbox and not managed by V8’s GC.
Implementation: These are implemented as indices into the ExternalPointerTable (EPT), which is located outside of the sandbox. The EPT performs type checks on every access to a pointer.

Conceptually, they are very similar to file descriptors in Unix. When the sandbox is disabled, they are regular full pointers.
Security Properties: The pointer will point to a valid object of the expected type. However, an attacker can swap these pointers as long as the type of the referenced object is the same.
Design Document: V8 Sandbox - External Pointer Sandboxing

Indirect Pointer

Summary: A pointer to a V8 HeapObject that goes through a pointer table indirection on access.
Implementation: The "under-the-hood" implementation of Trusted Pointers and Code Pointers (see below) when the sandbox is enabled. Similar to External Pointers, these are indices into a pointer table and perform type checks on access. These pointers are only available when the sandbox is enabled.
Security Properties: See Trusted Pointer.
Design Document: V8 Sandbox - Trusted Space

Trusted Pointer

Summary: A pointer that is guaranteed to point to a valid TrustedObject in trusted space.
Implementation: These are implemented as Indirect Pointers using the TrustedPointerTable (TPT) when the sandbox is enabled. Otherwise they are regular tagged pointers.
Security Properties: The pointer will point to a TrustedObject in trusted space of the expected type. However, as with External Pointers an attacker can perform "pointer swap attacks" so it is not guaranteed that the pointer still points to the "original" object.
Design Document: V8 Sandbox - Trusted Space

Code Pointer

Summary: A special kind of Trusted Pointer that always points to a Code object.
Implementation: Similar to Trusted Pointers, but these use the CodePointerTable (CPT) instead of the TPT. They are essentially a performance optimization as the CPT also directly points to the code's entrypoint.
Security Properties: Same as Trusted Pointers, but will always point to a Code object.
Design Document: V8 Sandbox - Code Pointer Sandboxing