# NaCl untrusted fault handling: guide to the implementation

## Overall

Non-host-OS-specific issues:

* [Issue 2401](http://code.google.com/p/nativeclient/issues/detail?id=2401): Overall tracking issue for the feature
* [Issue 2602](http://code.google.com/p/nativeclient/issues/detail?id=2602): Make sure NaCl’s fault-handling tests are run inside Chromium as well as in the standalone NaCl tests. This is particularly necessary given that we require Chromium-side code on Windows (see below).
* [Issue 2651](http://code.google.com/p/nativeclient/issues/detail?id=2651): Hook up fault handling in Chromium and plumb through an “enable\_exception\_handling” flag

## Status

The feature is currently present and working in Chromium/NaCl, but it is behind a flag: “--enable-nacl-exception-handling” must be passed to enable the feature. This is listed in “about:flags”.

## Tests

The test suite can be found in native\_client/tests/exception\_test.

## Windows implementation

The Windows implementation is the hairiest one since it uses the Windows debugging API. We have a “**debug exception handler**” thread which **runs outside of the NaCl loader process**.

* On x86-32, the Windows debugging API is the only way to catch untrusted-code crashes, because untrusted code runs with the %ss segment register modified. Windows normally kills the process if a crash occurs when %ss has a non-standard value.
* On x86-64, we could catch untrusted crashes by patching NTDLL, but since Windows does not have any equivalent of Unix’s sigaltstack(), the kernel will insist on writing a stack frame beneath %rsp. This means we wouldn’t be able to handle crashes due to stack exhaustion. Therefore, we use the debugging API on x86-64 as well.

Because the DebugActiveProcess() API takes a process ID, not a process handle, the debug exception handler must run outside the Windows Chrome sandbox. The architecture of the debug exception handler must match that of the NaCl loader process, so:

* On x86-32, the debug exception handler runs in the Chrome browser process (though, for robustness, we might want to move it into a separate process). I think this is the only code from the native\_client tree that runs in the browser process.
* On x86-64, the debug exception handler runs in the NaCl broker process (from nacl64.exe).

This was tracked under:

* [Issue 2536](http://code.google.com/p/nativeclient/issues/detail?id=2536): Implement for x86-64 Windows
* The first x86-32 implementation was committed in [r7409](http://src.chromium.org/viewvc/native_client?view=rev&revision=7409) without a tracking issue

The main part of the NaCl-side code is here:

* [native\_client/src/trusted/service\_runtime/win/debug\_exception\_handler.c](http://src.chromium.org/viewvc/native_client/trunk/src/native_client/src/trusted/service_runtime/win/debug_exception_handler.c?view=log)

The main Chromium-side code is:

* chrome/common/nacl\_debug\_exception\_handler\_win.cc
* there are code paths for invoking this on x86-32 and x86-64

Hairy parts:

* [Issue 2772](http://code.google.com/p/nativeclient/issues/detail?id=2772): NaCl's debug exception handler breaks Chromium's use of int3 to exit
* [Issue 2618](http://code.google.com/p/nativeclient/issues/detail?id=2618): We want to have the debug exception hander attached on demand (i.e. only when untrusted code is using fault handling) because it imposes a performance penalty. Every time a thread is created or exits, the whole process gets suspended.
* [Issue 2695](http://code.google.com/p/nativeclient/issues/detail?id=2695): We had to make sure this works even if chrome.dll gets relocated, which tends to happen on x86-32 Windows XP
* [Issue 2610](http://code.google.com/p/nativeclient/issues/detail?id=2610): We will have to make sure that Breakpad crash reporting doesn’t get broken
* [Issue 2572](http://code.google.com/p/nativeclient/issues/detail?id=2572): WriteProcessMemory() bypasses page permissions, so we have to use it carefully when writing a stack frame to the sandbox’s address space
* [Issue 2639](http://code.google.com/p/nativeclient/issues/detail?id=2639): The Windows debugging API is strange and assumes thread-local state, so it’s not obvious how it works when debugging multiple processes simultaneously

Possible future work:

* We might be able to make the debug exception handler work inside the Chrome sandbox by using internal Windows APIs that use process handles instead of process IDs.

## Linux implementation

The Linux implementation uses POSIX signal handling to catch untrusted faults. Our trusted fault handler **runs on the same thread** that produced the fault.

Implemented for x86-32, x86-64 and ARM.

This was tracked under:

* [Issue 2537](http://code.google.com/p/nativeclient/issues/detail?id=2537): Implement for Linux

The main part of the code is here:

* [native\_client/src/trusted/service\_runtime/posix/nacl\_signal.c](http://src.chromium.org/viewvc/native_client/trunk/src/native_client/src/trusted/service_runtime/posix/nacl_signal.c?view=log)

Hairy parts:

* [Issue 2397](http://code.google.com/p/nativeclient/issues/detail?id=2397): On x86-32 Linux, untrusted code can cause the trusted signal handler to run with insufficient stack
  + This behaves differently on Goobuntu kernels but I haven’t yet worked out why.

## Mac OS X implementation

The Mac implementation uses Mach exception handling, and so the trusted fault handler **runs on a separate thread** from the thread that produced the fault, but (for simplicity) runs in the same process.

In principle, we could have used POSIX signal handling on Mac, which would have allowed us to reuse the Linux implementation. But there were a couple of reasons to use Mach exception handling:

* Mac Breakpad uses Mach exception handling, and we need NaCl’s untrusted fault handler to coexist with Breakpad.
* In principle, Mach exception handling is more robust than POSIX signal handling, because the handler runs on a separate thread. This means we don’t need to worry about the handler inheriting context from untrusted code (e.g. an untrusted stack, the x86 direction flag).

Implemented for x86-32 only (but we don’t deploy NaCl’s x86-64 sandbox on Mac currently).

This was tracked under:

* [Issue 2538](http://code.google.com/p/nativeclient/issues/detail?id=2538): Implement for Mac

The main part of the code is here:

* [native\_client/src/trusted/service\_runtime/osx/mach\_exception\_handler.c](http://src.chromium.org/viewvc/native_client/trunk/src/native_client/src/trusted/service_runtime/osx/mach_exception_handler.c?view=log)