Tor Project
Tor Browser Bundle

Research Engagement

iSECpartners
part of nccgroup

Prepared for:

Prepared by:

Tom Ritter — Principal Security Engineer

Andy Grant — Principal Security Engineer

Table of Contents

1 Executive Summary

1.1 Project Summary

Open Technology Fund (OTF) engaged iSEC Partners for work with the Tor Project to evaluate Tor Browser Bundle. After discussions with Mike Perry at Tor Project, it was determined that the best use of time would be to conduct a more research-oriented engagement, looking at how exploitation may be made more difficult on Tor Browser Bundle, aiming to provide recommendations for an upcoming "Security Slider" feature.[1]

Note: Tor Browser Bundle is based on the Firefox browser. In this document, iSEC has used "Tor Browser Bundle" when it is speaking specifically about the browser distributed by the Tor Project, and "Firefox" when speaking about features that apply to both distributions.

The Security Slider will aim to disable certain features of Tor Browser Bundle at higher levels of security. To this end, iSEC was granted access to many private bugs on the Mozilla bug tracking software to catalog past vulnerabilities of Firefox by type and component. During this process, iSEC also analyzed several public and private exploits against Tor Browser Bundle and Firefox to investigate if there were any significant commonalities that could guide hardening recommendations.

Firefox has a robust set of preferences for controlling features through the `about:config` interface. Several preferences relevant for the security slider are enumerated later in this report. While many of the features Tor Project may wish to disable or control are exposed through these settings, many are not. Therefore, iSEC examined different approaches to add these settings to the codebase, and developed patches in certain instances.

iSEC also looked at more general hardening options that can be made to Tor Browser Bundle. Compiler settings that include strict memory checks are being explored by the Tor Project already, and include building Tor Browser Bundle with Address Sanitizer[2] - two items that can be added to this list are the Windows setting `EnableTerminationOnHeapCorruption` and an experimental feature in GCC named Virtual Table Verification. Additionally, iSEC confirmed that Address Space Layout Randomization, a best-practice feature for making exploitation more difficult, is currently omitted on Windows and Mac builds.

Another general hardening option iSEC investigated was replacing Tor Browser Bundle's memory allocator, jemalloc, with a hardened allocator. PartitionAlloc,[3] developed by the Chrome Security team appears to be a good base for improving security through its feature-set.

Several other tasks were performed, including suggesting ways to detect regressions in exposed DOM objects that may aid in user fingerprinting, and developing patches to enable assertions in specific critical components.

[1] https://trac.torproject.org/projects/tor/ticket/9387
[2] https://trac.torproject.org/projects/tor/ticket/10599
[3] https://chromium.googlesource.com/chromium/blink/+/master/Source/wtf/PartitionAlloc.h

1.2 Recommendations Summary

Browsers have evolved in complexity tremendously over the past decade, and the Tor Project is in a very difficult situation with regards to it. Their ultimate goals of preventing fingerprintability and proxy leaks are not universally shared by Mozilla and the Tor Project development team is much smaller. The aggressive release of Firefox versions is offset by their Extended Support Releases, but this still necessitates a large evaluation of new features and patch-reconfiguring every 10 months. Furthermore, the Tor Project is in the process of developing significant features on top of Tor Browser Bundle - the new Tor Launcher, automatic updates, and the Security Slider.

In short, the road Tor Project is embarking on will be difficult to continue while maintaining high security standards without considerable cooperation with Mozilla, a sustainable development group, and periodic involvement from specialized individuals.

Short Term

For the purpose of this research document, short-term recommendations are meant to be undertaken on the 1-6 month timeline. While all recommendations in this report are longer term in relation to typical vulnerability remediation, this area is a summary of strategic recommendations that should be taken in the short term to guide development efforts and protect users.

Re-enable Address Space Layout Randomization on Windows and Mac builds. Currently Tor Browser Bundle builds for Windows or Mac do not have ASLR enabled universally. ASLR is a best-practice for browsers, and omitting it makes it significantly easier for attackers to bypass the (currently enabled) Data Execution Prevention settings. In addition to re-enabling ASLR, develop regression tests that ensure that ASLR is enabled on all future builds.

Participate in the "Pwn2Own" Contest. Speak with the sponsors of the Pwn2Own and Pwnium contests, and see if they would be willing to allow the Tor Project to participate. Because Tor Browser Bundle is based on Firefox, change the target by attempting to standardize on a 'Medium' Security Level, which replaces the memory allocator with PartitionAlloc, disables significant functionality (such as Web Fonts and SVG) but leaves JavaScript enabled. Stabilize this selection in the Fall, several months before the contest, and change the goal from 'system compromise' to demonstrating a proxy bypass. (This will have the added benefit of allowing someone to claim a prize by demonstrating a bypass that does not achieve exploitation.) Review the exploitation techniques used, and depending on outcome, consider raising the difficulty to a 'High' security slider setting for the following year.
Note that this recommendation is a short-term recommendation primarily because of the time of year - if Tor Project moved quickly on this, it would potentially be possible to participate in 2015 contest coming up.

Test Windows Firefox Exploits with Microsoft EMET. The Enhanced Mitigation Experience Toolkit (EMET),[4] currently at version 5.0, is a Microsoft-provided application that adds additional exploit mitigations to try and detect and defeat certain exploitation techniques. It is not perfect, but it is currently unknown if it would have prevented any actual exploit attempts on Firefox. Depending on its usefulness, it may be worth recommending to Windows users.
Note: This may only be possible for Mozilla to do, unless the exploit examples are provided to the Tor Project.

Long Term

For the purpose of this research document, long-term recommendations are meant to be undertaken in the 6 month and beyond timeline. These may include significant changes to the architecture or code and may therefore require in-depth planning, complex testing, significant development time, or changes to the user experience that require retraining.

Note: Many of the recommendations that iSEC would ordinarily make, such as developing an automatic and secure update mechanism, are already being developed by the Tor Project. These recommendations are omitted in the name of redundancy. Similarly, many recommendations, such as process sandboxing, are large and ambitious and probably outside the Tor Project's current capability.

Closely follow the Chrome Security team. The Chrome Security team has been a source of innovation in the browser security space. Tor Browser Bundle is based on Firefox and thus inherits progress made by Mozilla automatically. While improvements in Chrome may not be appropriate for Firefox, they could be integrated in Tor Browser Bundle. In a best case scenario, members of the Chrome Security team may be allowed to work with the Tor Project on these changes.

Replace the jemalloc allocator with ctmalloc and partition object allocation types. PartitionAlloc, used by ctmalloc, removes in-line heap metadata and when used with separate partitions isolates object types. When used to its full capabilities, it should be considerably more hardened than jemalloc. This should make exploiting common heap corruption vulnerabilities more difficult.

Investigate strategies to harden against Use After Free (UAF) exploits. A significant number of exploits and vulnerabilities that iSEC reviewed are Use After Free vulnerabilities. More recent versions of GCC seem to have some support for the 'final' keyword and Virtual Table Verification, which are two possible mitigations. Another area of investigation is using the partitioning features of PartitionAlloc to separate DOM objects from user-controlled buffers like strings and arrays. Future research efforts could be conducted by the Tor Project, affiliated or unaffiliated groups, to make improvements in this area.

Develop a Firefox ESR migration process. Upgrading between Firefox ESR versions introduces a considerable amount of features being added to the browser, and additional preferences being enabled that previously were off by default. Using the techniques described in section 3.9 on page 26 and section 3.10 on page 26, develop a plan for migrating between ESR releases that includes a wiki page that individuals can contribute to for tracking added functionality to Firefox.

[4] https://connect.microsoft.com/directory/?keywords=EMET

2 Engagement Structure

2.1 Internal and External Teams

The iSEC team has the following primary members:

- Andy Grant — Principal Security Engineer
  agrant@isecpartners.com
- Tom Ritter — Principal Security Engineer & Account Manager
  tritter@isecpartners.com

The Tor Project team has the following primary members:

- Mike Perry — Tor Project
  mikeperry@torproject.org

2.2 Project Goals and Scope

The goal of this engagement was to determine what techniques could be used to harden Tor Browser Bundle against attacks in default and user-selected higher security modes. This included:

- Reviewing Tor Browser Bundle's use of compiler and OS-specific hardening options
- Investigating enabling debug assertions in production releases
- Reviewing past exploitable bugs in Firefox to determine their type, origin, and what components (if any) could have been disabled to prevent exploitation
- Identify and enumerate audio and video parsing libraries in use by Firefox
- Identifying and reviewing protocol handlers enabled in Tor Browser Bundle
- Review `about:config` settings and components in Firefox that are unneeded or represent significant sections of code that can be disabled

3 Detailed Research Findings

3.1 Bug Classification

iSEC begun classifying the private bugs that related to the ~70 CVEs Firefox has had since Firefox 24.[5] The issue type and affected component is primarily determined from Mozilla's classification and comments on the issue, an explanation of the terms used can be found in Appendix A on page 29, and components with only a single issue are omitted.

Component	
JS Core	29
Ion	24
DOM Core	19
Networking	6
WebRTC	5
WebGL	5
Undetermined	5
asm.js	4
ImageLib	4
Web Audio	2
SVG	2
IndexDB	2
Image	2
Editor	2
Dom Core	2
DOM Sore	2
Canvas 2D	2
Audio	2

Vulnerability Type	
UAF	43
Undetermined	35
Assert	28
UUIM	6
Null Deref	3
Heap Overwrite	3
Stack Buffer Overwrite	2
Integer Overflow	2
Data Leak	2
Type Confusion	1
Stack Overflow	1
Memory Leak	1
Heap Overread & Overwrite	1
Heap Overread	1
Double Free	1

[5] Specifically, iSEC reviewed the bugs linked to by the Mozilla Foundation Software Advisories from Sept 17, 2013 to April 29, 2014.

iSEC also began to review public bugs suggested by Mozilla, using a specific query.[6] These issues are largely from the mid-2013 timeframe, and are skewed towards the Web Audio category, as it seems to have had a large category change. This second table does not represent a complete view of data from a particular time period.

Component		**Vulnerability Type**	
Web Audio	18	UAF	14
JS Core	5	Heap Overwrite	8
SVG	3	Heap Overread	4
DOM Core	2	Assert	4
WebGL	1	Stack Pointer Corruption	1
Persona/Identity	1	Stack Buffer Overwrite	1
file:// URL	1	Undetermined	1
IndexDB	1		
ImageLib	1		

3.2 Exploit Analysis

iSEC analyzed four exploits for Firefox and Tor Browser Bundle that were discovered in the wild, documented publicly, or provided by Mozilla. Exploit analysis can indicate which techniques real-world attackers use to compromise browsers, and guides exploit mitigations. HP's Pwn2Own,[7] Google's Pwnium,[8] and Microsoft's Heart of Blue Gold[9] programs are all designed to understand how real-world exploits and exploit mitigations work, and how software can be hardened in effective ways.

Tor Browser Bundle shares a significant amount of attack surface with Firefox. However, currently there is a significant difference in threat model - it is absolutely critical for Tor Browser Bundle not to expose any proxy leaks that would send traffic outside the configured SOCKs proxy. In the future, as the Security Slider is developed and the memory allocator potentially replaced, Tor Browser Bundle will diverge even further from Firefox. iSEC recommends working with third parties to attempt to participate in these contests to gather intelligence on how well Tor Browser Bundle meets its specific goals and how attackers can circumvent hardening options Tor Browser Bundle incorporates.

It is likely that exploits against Firefox will continue to guide decision-making for Tor Browser Bundle and the Security Slider, analyzing these exploits now and in the future will continue to be important.

August, 2013 Freedom Hosting Exploit

The Metasploit team performed an analysis of the exploit,[10] which says it uses an information leak to craft a ROP chain specifically for Windows 7 using ntdll, and transfers execution into that chain using

[6] https://bugzilla.mozilla.org/buglist.cgi?j_top=OR&f1=keywords&o1=anywordssubstr&resolution=---&resolution=FIXED&classification=Client%20Software&classification=Components&o2=anywordssubstr&query_format=advanced&f2=status_whiteboard&v1=sec-high%20sec-critical&v2=sg%3Ahigh%20sg%3Acritical&list_id=10101000

[7] http://www.pwn2own.com/

[8] http://blog.chromium.org/2014/01/show-off-your-security-skills.html

[9] http://blogs.technet.com/b/bluehat/archive/2013/06/19/heart-of-blue-gold-announcing-new-bounty-programs.aspx

[10] https://community.rapid7.com/community/metasploit/blog/2013/08/07/heres-that-fbi-firefox-exploit-for-you-cve-2013-1690

a stack pivot also in ntdll. The ROP chain calls `ntdll!ZwProtectVirtualMemory` to disable DEP and then moves into the exploit payload.

Good analyses of the exploit's payload were conducted by Gareth Owen[11] and Vlad Tsyrklevich.[12] The payload has a few interesting points. Firstly, it uses a function resolver included in Metasploit[13] to identify where functions it wishes to call are in memory. Secondly, it loads two libraries `iphlpapi.dll` and `ws2_32.dll` - the second library contains a `connect()` call the payload uses to send a request, the first contains the `SendARP()` function the payload uses to determine the system's MAC address. The running instance of Tor Browser Bundle already has functions that can be used to issue requests (eliminating the need for `ws2_32.dll`). It is unknown if there is an existing function that could obtain the system's MAC address, but it seems likely.

VUPEN 2014 Pwn2Own

This analysis is based on VUPEN's writeup at the following URL:
http://www.vupen.com/blog/20140520.Advanced_Exploitation_Firefox_UaF_Pwn2Own_2014.php

In the Pwn2Own content in 2014, VUPEN exploited a Use After Free vulnerability that resulted by Firefox being placed into a 'memory-pressure' state. The object itself was not a DOM object or other object created by the webpage, but rather a "BumpChunk" object that is created by the allocator for managing memory.

After the BumpChunk is freed, VUPEN creates an ArrayBuffer in its place, which is manipulated to gain read and write access to the entire process address space. With read access, the exploit can defeat ASLR, and build a ROP chain using mozjs.dll.

There are a few interesting components of the exploit. They exploited the memory-pressure state of Firefox, but not for any unique properties of that state but rather because entering that state caused a Use After Free itself. Through clever manipulation of the ArrayBuffer and View, VUPEN was able to create an ArrayBuffer with length 0x01000000, which is large enough to edit a second ArrayBuffer with length 0xFFFFFFFF, which in turn can read and write to any location in the process address space.

Private Exploits

iSEC also analyzed exploits that were submitted privately to Mozilla. Interesting characteristics about these exploits were:

- Several exploits use ArrayBuffers with invalid lengths, and one used a technique very similar to VUPEN's, creating an ArrayBuffer and then a view with an invalid length that was used to write into arbitrary memory.
- Another exploit used a vulnerability that allowed the author to execute JavaScript as the system principal (in the Firefox use of the phrase, not a root or SYSTEM user account) achieving arbitrary code execution. Most notably, this exploit did not use any memory corruption to achieve code execution.

[11] http://ghowen.me/fbi-tor-malware-analysis/
[12] http://tsyrklevich.net/tbb_payload.txt
[13] https://github.com/iagox86/nbtool/blob/master/samples/shellcode-win32/block_api.asm

3.3 Security Slider Thoughts

This section contains individual components of Firefox that iSEC has researched either through existing preference settings or bug categories. iSEC's recommendations are based around the following Security Levels.

- **None** - TBB is configured in its most permissive state
- **Low** - High-Risk components are disabled, unless they are used by a large percentage of websites
- **Medium** - High-Risk components are disabled unless they are used by an overwhelming majority of websites. Medium-Risk components are disabled, unless they are used by a large percentage of websites.
- **High** - JavaScript is disabled. Many if not most components are disabled in the name of reducing attack surface.

media.webaudio.enabled

The Web Audio feature is disabled in Firefox 24 and Tor Browser Bundle. It was enabled in Firefox 25[14] and is now on by default. After reviewing security-relevant bugs in Firefox, a significant number of potential vulnerabilities were found in this component.

Recommendation: Disable at the Low or Medium security level.

media.audio_data.enabled

The Audio API was an experimental API superseded by the Web Audio API.[15] In Firefox 24 and Tor Browser Bundle it was enabled, but is disabled in Firefox 28.

Recommendation: Disable at the Low security level.

layout.css.flexbox.enabled

This preference has been true by default since Firefox 22, and the preference itself was removed in Firefox 28.[16] iSEC does not have a specific recommendation for this setting, but wanted to note that the revision that removes the preference is at https://hg.mozilla.org/mozilla-central/rev/1a09d295aa1c, and is simple enough that it may be re-added, or potentially copied to other styles.

gfx.downloadable_fonts.enabled

Web Fonts in .ttf, .otf, and .woff formats can be downloaded, parsed, and used by Tor Browser Bundle by default. Mozilla conducted a Security Review of downloadable fonts,[17] and their concern was the same as ours: that the font parsing subsystems could have vulnerabilities that an attacker could exploit. To mitigate this threat, Firefox integrates the OpenType Sanitizer.[18]

[14] https://developer.mozilla.org/en-US/Firefox/Releases/25#Interfaces.2FAPIs.2FDOM
[15] https://developer.mozilla.org/en-US/docs/Introducing_the_Audio_API_Extension\protect\char"0024\relaxhistory
[16] https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes
[17] https://wiki.mozilla.org/Firefox3.1/Downloadable_Fonts_Security_Review
[18] https://code.google.com/p/ots/

The OTS Sanitizer appears to be effective at preventing exploitable bugs. No software is perfect however, and there is a lot of concern around Font Parsing on Windows.[19]

Recommendation: Disable at the High security level. Ordinarily, iSEC would recommend disabling these at the Low or Medium security level, but the Tor Browser Bundle team has indicated that they wish to prefer remote fonts over local fonts for user fingerprinting reasons.

gfx.font_rendering.graphite.enabled

The Graphite Font Shaping feature[20] is functionality used to more accurately render complex scripts in South-East Asian dialects. The feature has been enabled by default since approximately Firefox version 12.

At least one security-relevant bug in the last year (836225) was found in graphite parsing, as well as three in the last two years (752662, 753230, and 753623 which is CVE-2012-3971). iSEC believes this is indicative of other issues present in the code base. The library is not maintained by Mozilla, and while Mozilla indicates they fuzz it, it is not clear how often with respect to new releases, or how thoroughly. It was subject to a security review by Mozilla.[21]

Recommendation: For South-East Asian or other relevant locales, disable at the Medium or High security level. For other locales, disable at the Low security level.

gfx.font_rendering.opentype_svg.enabled

SVG in OpenType fonts is a featured designed to provide support for using SVG inside font files to create colored, animated, or more expressive glyphs in fonts.[22] In Firefox, this feature was disabled in ESR 24, and is enabled in (at least) Firefox 29. iSEC was unable to find any security review of this feature, or security-relevant bugs. iSEC does not expect high usage of this feature on the Internet, as it does not appear to be supported in any other browsers - a competing solution, SVG fonts,[23] is implemented in Chrome, Safari, and Opera.

Recommendation: Disable at the Low security level.

media.*.enabled

As explained in section 3.7 on page 23, there are several codecs used or enabled in Tor Browser Bundle, and each have seen security vulnerabilities at the Critical level and below. iSEC was unable to make a determiniation if any formats were used more or less commonly on the web that could guide a decision to disable one or more of these features at the Low security level.

Recommendation: Disable at the Medium security level.

[19] http://threatpost.com/of-truetype-font-vulnerabilities-and-the-windows-kernel/101263

[20] https://wiki.mozilla.org/Features/Platform/Graphite_font_shaping, http://scripts.sil.org/cms/scripts/page.php?site_id=projects&item_id=graphite_fontdemo

[21] https://wiki.mozilla.org/Security/Reviews/Firefox/Graphite

[22] More information can be found at http://robert.ocallahan.org/2013/02/svg-in-opentype-new-approach-to-svg.html, http://robert.ocallahan.org/2013/08/svg-in-opentype-progress-update.html, https://wiki.mozilla.org/SVGOpenTypeFonts, and https://bugzilla.mozilla.org/show_bug.cgi?id=719286

[23] http://caniuse.com/svg-fonts

dom.indexeddb.enabled

The IndexedDB feature is currently disabled in Tor Browser Bundle for user fingerprinting reasons.[24] In addition to these reasons, iSEC would like to raise concerns with its security, as there is a small history of security vulnerabilities in the feature. Although Mozilla has conducted a security review,[25] its complex featureset and API imply a large and complex codebase where vulnerabilities may reside.

Recommendation: Continue to disable at the 'None' or Low security level.

javascript.options.asmjs

This setting controls the ASM.js feature in Firefox. Disabling this function will still allow JavaScript execution, but it will not be performed by the more optimized ASM.js engine. A few bugs have been present in the ASM.js codebase, but because of its constrained environment, exploitation may require more tricks as many of the common exploit techniques may not apply.

Recommendation: Disable at the Medium security level.

Ion JIT Compiler and Related Options

At the request of the Tor Project, iSEC investigated three settings related to the newer Ion JIT Compiler:

- javascript.options.ion.content
- javascript.options.baselinejit.content
- javascript.options.typeinference

Ultimately, while disabling these features will remove code paths with a history of vulnerabilities - the public exploit pattern seems to be more focused around Use After Free vulnerabilities, and thus it does not seem it will remove code paths attackers actually target for exploitation. Additionally, iSEC understands that are user reports of having these settings disabled and experiencing poor performance, which much also factor into the decision.

Recommendation: Disable at the Medium security level.

webgl.disabled

WebGL is a JavaScript API for rendering interactive 2D and 3D graphics in the `<canvas>` element. In 2014 alone, it has been the source of 3 sec-critical, 3 sec-high, and 1 sec-moderate bugs in Mozilla's bugtracker.

Recommendation: Disable at the Low or Medium security level.

jar: protocol

As explained in section 3.8 on page 25, the `jar:` protocol handler is a Firefox-specific feature that is largely unused on the broader Internet, mostly being used in Intranet sites. Its unusual nature, moderate complexity, and lack of widespread use make it a strong candidate for disabling.

[24] https://trac.torproject.org/projects/tor/ticket/8382
[25] https://wiki.mozilla.org/Security/Reviews/Firefox4/IndexedDB_Security_Review

Recommendation: Disable at the Low security level using the supplied patch.

SVG

The SVG components have been the host of several exploitable bugs in the past several years. Unfortunately, Firefox does not have a built-in preference to disable SVG, as it was removed[26] when it was determined that Firefox itself used SVG internally, and thus the preference could not be supported. iSEC did not have time to investigate if SVG could be easily removed - an initial search yielded a potential function in `content/svg/content/src/nsSVGFeatures.cpp`, but this function does not control functionality and merely reports an answer for the `document.implementation.hasFeature` functionality check.

Recommendation: Disable at the Low or Medium security level.

JavaScript

Clearly there are a number of bugs that fall into the JavaScript Core component. These bugs would be difficult to eliminate without entirely disabling JavaScript, which is required for most of the Web to function.

Recommendation: Disable at the High security level.

TLS Settings

Most web browsers, including Firefox, do not have as strict settings on TLS as may be desired in certain situations. The Tor Project could consider preventing the use of RC4, removing protocol downgrades to TLS versions below TLS 1.2 or 1.1, requiring DHE ciphersuites, removing the option to click through self-signed certificates, or removing certain Certificate Authorities from the trust store. Revocation presents an interesting situation: on the privacy side there is an argument to disable remote OCSP queries to avoid leaking this data to a third party; but on the security side there is an argument for enforcing OCSP Hard Fail.

[26] https://bugzilla.mozilla.org/show_bug.cgi?id=617448

3.4 Compiler Hardening

Microsoft Windows

iSEC investigated how the gitian build system compiled Tor Browser Bundle for Windows. While Mozilla builds Firefox using Microsoft Visual Studio compilers, gitian uses MinGW to compile Tor Browser Bundle using gcc on Linux targeting Windows. This affects many of the exploit mitigation technologies that are used on Windows.

The `-fstack-protector-all` (or `-fstack-protector-strong`) options should be used to protect against stack-buffer overflows. Comments in `descriptors/gitian-firefox.yml` indicate that this setting is currently disabled.

Examining the process in Process Explorer[27] revealed that Tor Browser Bundle *does* have Data Execution Prevention (DEP) enabled, but it does not universally enable Address Space Layout Randomization (ASLR). The following components *do not* have ASLR enabled as of Tor Browser Bundle 3.6.1:

1. browsercomps.dll*
2. firefox.exe*
3. freebl3.dll*
4. gkmedias.dll*
5. mozalloc.dll*
6. mozglue.dll*
7. mozjs.dll*
8. mozsqlite3.dll
9. nspr4.dll
10. nss3.dll*
11. nssckbi.dll*
12. nssdbm3.dll*
13. nssutil3.dll
14. plc4.dll
15. plds4.dll
16. smime3.dll
17. softokn3.dll*
18. ssl3.dll
19. xul.dll*

Note: Items marked with a * are present in the vanilla Firefox ESR and are marked ASLR there. Items without a * are not present in the vanilla Firefox ESR distributable. The pefile python module, and the script located at http://security.stackexchange.com/questions/43681/how-can-i-detect-or-inventory-all-dlls-that-dont-use-aslr, can be used to check if ASLR is enabled programmatically.

Also of note is that Firefox and Tor Browser on Windows are both 32-bit applications. The limited address space provided by 32-bit applications allows a good degree of confidence in exploits that spray the heap. a 64-bit build of the browser, combined with comprehensive ASLR, would make these exploits extremely unreliable.

iSEC used `dumpbin.exe /loadconfig` (provided with Microsoft Visual Studio Express) to check if firefox.exe or the supporting dll's were compiled with SafeSEH,[28] and determined that in Firefox ESR they are, but in Tor Browser Bundle they are not. While investigating exception handling implementations, iSEC determined that when gcc is used to cross-compile for Windows, gcc does not implement Structured Exception Handling, instead using "setjmp/longjmp"-based exception handling.[29]

However, when Firefox is compiled with gcc, it explicitly disables exception handling with the `-fno-exceptions` option. This appears to be intended only for Linux builds, but Tor Browser Bundle inherits

[27] http://technet.microsoft.com/en-us/sysinternals/bb896653

[28] Windows also provides the SEHOP option to harden against SEH exploitation; however, this is not a compiler option, and instead must be opted into via the Windows Registry: http://blogs.technet.com/b/srd/archive/2009/11/20/sehop-per-process-opt-in-support-in-windows-7.aspx.

[29] http://gcc.gnu.org/wiki/WindowsGCCImprovements

the setting for Windows as well. iSEC believe that both Structured Exception Handling and setjmp-longjmp-based exception handling are missing from gcc-compiled code, but is uncertain if other Windows mechanisms may place exception handlers on the stack.

In "ipc/chromium/src/base/process_util_win.cc" Firefox sets `EnableTerminationOnHeapCorruption`,[30] but this function does not seem to actually be called except in a test suite. `EnableTerminationOnHeapCorruption` applies to user-mode heaps created by `HeapCreate()` (which is called in "sqlite3.c" and has matches in "CityHash.dll" and "ApplicationID.dll") and the process heap (obtained by `GetProcessHeap()` and called in a few places in the codebase). According to Microsoft,[31] this setting has no impact on performance, so it is probably worth enabling.

gcc has an experimental Virtual Table Verification feature.[32,33] This feature must be compiled into gcc which is unusual, but Tor Browser Bundle's deterministic build system already compiles gcc from source - however the feature is not in the gcc 4.6 branch, which is what Tor Browser Bundle uses currently. VTV aims to limit exploitation of Use After Free vulnerabilities by protecting the vtables of C++ objects. UAF accounts for a significant number of vulnerability types, and a significant number of exploitation vectors actually used in the wild. Integrating this could be very worthwhile.

Another technique to mitigate UAF vulnerabilities is to reduce the number of vtable lookups, as these lookups often lead to code execution. If the class does not look up function pointers from attacker-controlled heap memory, the risk of code execution is reduced. Classes that are not overridden can be automatically marked 'sealed' or 'final', and their vtable calls turned into direct calls, also yielding a small performance improvement. Microsoft has performed this optimization on certain libraries in Internet Explorer.[34]

Update: Following discussions after the engagement, iSEC determined that Clang[35] and gcc as of 4.9[36] also support this feature in some manner. It will be necessary to investigate gcc's behavior more carefully to determine how to make use of it (for example, if the final attribute can be added automatically).

One final technique that is used in Chromium to mitigate UAF exploitation is separate heaps for DOM objects and strongly user-controlled objects like strings and vectors. PartitionAlloc separates these types of objects into different heaps.

Apple OS X

iSEC verified that Tor Browser Bundle on OS X has a non-executable stack (NX, also known as DEP on Windows) by checking that the threads' stacks have their permissions set to `rw-` using the `vmmap` tool.

iSEC also checked the ASLR status using `otool -hv` on the firefox binary distributed in the Tor Browser Bundle App, and determined that it is lacking the PIE attribute - lacking the attribute opts the application out of ASLR on OS X. While reviewing the differences between the Tor Browser Bundle build process and Mozilla's, iSEC discovered that both Tor Browser Bundle and Firefox are built with the 10.6 SDK. The primary difference is that Firefox is built with `-arch x86_64` while Tor Browser Bundle is

[30] http://blogs.msdn.com/b/oldnewthing/archive/2013/12/27/10484882.aspx
[31] http://msdn.microsoft.com/en-us/library/bb430720.aspx
[32] https://gcc.gnu.org/wiki/vtv
[33] Microsoft Visual C++ Compiler has a feature called "vtguard" that provides similar functionality.
[34] http://media.blackhat.com/bh-us-12/Briefings/M_Miller/BH_US_12_Miller_Exploit_Mitigation_Slides.pdf
[35] http://stackoverflow.com/questions/7538820/how-does-the-compiler-benefit-from-cs-new-final-keyword
[36] http://gcc.gnu.org/gcc-4.9/changes.html

built with `-arch i386`. Changing this setting should enable ASLR on OS X, as the ASLR in 10.6 is not applicable to x86 applications.

However, the ASLR in OS X 10.5 and 10.6 (it was not upgraded in 10.6) is ineffective. It does not randomize the position of system libraries, only application libraries - so building ROP chains is still trivial thanks to the fixed addresses. It is not necessary to build with the 10.7 SDK once PIE is enabled, as the improved ASLR will take effect automatically on OS X version 10.7 and above, but it is important to note that OS X 10.6 and below are significantly less secure in this regard.

While reading the build-helper scripts for OS X, iSEC noticed there are several typos in the `-DMAXOSX_-DEPLOYEMENT_TARGET` option. To be used for its predefined purpose, this option should be `MACOSX_-DEPLOYMENT_TARGET`[37] (MAC instead of MAX, and remove the extra 'E' in deployment.) Currently, this option has no effect, as the default deployment target if unset is the version of the SDK used (which is also 10.6).

AppArmor Sandbox

iSEC briefly read a provided `local.tbb3.apparmor` policy file, but did not have time to iterate on it or investigate the many permissions that are granted but commented for later review - these include allowing UDP packets and full tcp network access instead of only to 127.0.0.1.

iSEC did notice that, through `#include <abstractions/dbus-session>`, access is to granted to the machine-unique identifier in the `/var/lib/dbus/machine-id` file. The man page for the `dbus-uuidgen` tool indicates that it should be able to be regenerated at every machine reboot.

[37] https://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/cross_development/Configuring/configuring.html

3.5 Enabling Assertions

iSEC spent some time looking at assertions within Tor Browser Bundle and the feasibility of enabling them in non-debug builds. The first pass of this involved modifying the system's assert.h file, replacing the line `#ifdef NDEBUG` with `#ifdef TOR_NASSERT`. This causes assert.h-based assertions to exist in non-debug builds. Minor code changes were required to address compilation errors. Most notably, sqlite3 had excessive compilation errors, likely due to its custom debug defines. As such, sqlite3 was changed to compile against an unmodified assert.h. The only other changes were in the libnestegg and dwarf libraries and required one change each to define a normally debug-only variable. See Appendix E.1 on page 47 for a sample of the patch to enable system asserts.

After the successful compilation and execution of Tor Browser Bundle with assert.h-based assertions enabled, iSEC reviewed the Mozilla code for custom assertions. There were numerous custom assertion-type functions, largely defined in tor-browser/xpcom/glue/nsDebug.h. An attempt to enable these assertion methods resulted in a multitude of compilation errors. Similar to the errors seen when enabling the system assertions, these largely were due to debug-only variables and functions not being defined for use in the assertion function. Some time was spent trying to address these issues but it was determined that resolving all of them to make the browser buildable would likely take too much amount of time to complete successfully.

While many situations are easily rectified using the `DebugOnly<T>` templated class, there are corner cases of variable assignment that would have to be tracked down.

Instead of attempting to enable all assertions, enabling asserts in targeted classes was revisited with a focus on historically-vulnerable components. This included the reference counting classes of `nsCOMPtr` and `nsRefPtr` as well as the JavaScript engine. Enabling the Mozilla-based assertions within the reference counters was straightforward and had no apparent side effects. See Appendix E.2 on page 49 for a sample patch. Similarly, the Mozilla-based assertions were enabled in the JavaScript code with minimal complications. Upon initially building Tor Browser Bundle and performing basic web browsing, one of the JavaScript assertions was triggered. This was due to a missed debug-only function declaration but acted as validation that the assertions were being enabled. The JavaScript engine has its own set of assertions but enabling them proved more difficult with many more corner cases to hunt down. iSEC was successful in compiling the browser with JS assertions enabled, but the browser regularly crashes from failed assertions, most likely caused by missing debug variable declarations. See Appendix E.3 on page 59 for a sample of the latest patch.

3.6 Memory Allocator Replacement

When exploiting memory corruption, one of the most important things to understand and manipulate is the application's memory allocator. Firefox's memory allocator is jemalloc, and it has been the subject of study for exploitation purposes [38,39,40,41] for Firefox and other open source projects that use it.

Another popular memory allocator is TCMalloc, which is used in WebKit, and therefore Chrome, Safari, Android, BlackBerry and many other pieces of web browsing software. TCMalloc has also been the target of study for exploitation purposes,[42] and while very fast, does not provide as much security as other allocators.

Google has recently created a new allocator for Blink named PartionAlloc[43] that was written with speed and security in mind. In particular, one of the mechanisms it uses to achieve more security is by using different memory arenas ('Partitions') for different types of allocations, for example rendering, buffering, and certain object models. Of note, they separate DOM objects from ArrayBuffers and strings, which makes Use After Free vulnerabilities more difficult to exploit.[44]

Because PartitionAlloc requires a partition choice, a new generic allocator, named ctmalloc,[45] is in development for Chromium. ctmalloc uses PartitionAlloc on the backend, and places all allocations into a single Partition when called through the standard malloc()/free() interface. While this is simple, it does not provide all of the intended security benefits of ParitionAlloc. Furthermore, Firefox's use of malloc, and the malloc replacement API, do not easily lend themselves to explicitly choosing a partition. One idea offered by PartitionAlloc's developer was to create a number of partitions and segment allocations into those partitions based on a per-execution secret and the allocation location (from EIP).

Overridding

Swapping out the memory allocator in Firefox is not a trivial process. Fortunately, Mozilla already did it, and now it is as simple as building with "–enable-replace-malloc" and executing Firefox with

1. On GNU/Linux:
   $ LD_PRELOAD=/path/to/library.so firefox
2. On OSX:
   $ DYLD_INSERT_LIBRARIES=/path/to/library.dylib firefox
3. On Windows:
   $ MOZ_REPLACE_MALLOC_LIB=drive:\path\to\library.dll firefox

[38] BlackHat 2012: https://media.blackhat.com/bh-us-12/Briefings/Argyoudis/BH_US_12_Argyroudis_Exploiting_the_%20jemalloc_Memory_%20Allocator_WP.pdf and https://www.youtube.com/watch?v=7kgGVPhB2fk

[39] In Phrack: http://phrack.org/issues/68/10.html#article & http://phrack.org/issues/68/13.html#article

[40] OWASP AppSec: http://census-labs.com/media/heap-owasp-appsec-2012.pdf

[41] The Browser Hackers Handbook, http://books.google.com/books?id=lXr0AgAAQBAJ&pg=PT276&lpg=PT276&dq=exploiting+jemalloc&source=bl&ots=vdnwCXuuAD&sig=AB56x3njLjDh5OyV5Z8seOj2OXk&hl=en&sa=X&ei=x1FyU5LnMfbMsQTyyYHoCg&ved=0CDwQ6AEwBDgK#v=onepage&q=exploiting%20jemalloc&f=false

[42] http://immunityinc.com/infiltrate/archives/webkit_heap.pdf

[43] https://chromium.googlesource.com/chromium/blink/+/master/Source/wtf/PartitionAlloc.h

[44] http://nullcon.net/website/archives/download.php?filename=Chrome-OS-Security-2014-New-and-future-hotness-by-Sumit-Gwalani.pdf

[45] https://code.google.com/p/chromium/issues/detail?id=339604

The issue that tracks adding the feature is Bugzilla #804303[46] and an excellent blog post explaining how to use it is at `http://glandium.org/blog/?p=2848`.

iSEC successfully created a sample memory replacement library against Firefox ESR 24, the patch is included in Appendix F.1 on page 145.

Replacing with ctmalloc

iSEC used the `ctmalloc-0.0.2.tar.gz` release from the chromium project[47] as a base for building a malloc replacement library. While iSEC changed all ASSERT's in the files to RELEASE_ASSERT's for debugging purposes, the major adaptations took place in malloc.cpp, which is included in Appendix F.2 on page 148.

Using this library causes Tor Browser Bundle to crash in `sqlite3.c:sqlite3VdbeMakeReady` - debugging indicates this is because `growOpArray` will eventually call into `moz_malloc_usable_size`. The usable_size function is not overridden by ctmalloc, and thus goes into the jemalloc routines, which do not know about the pointer, and returns 0. This makes `nOpAlloc` 0, eventually causing the segmentation fault.

In the time allocated, iSEC did not have time to develop a usable_size function for ctmalloc, but the next steps for continuing this effort will be to do so. It will probably be necessary to override all malloc functions defined by the replace_malloc API.

Update: Following the engagement and conversations with PartitionAlloc's developer, iSEC used an updated version of PartitionAlloc that implements usable_size. This successfully compiled and ran Tor Browser using ctmalloc. Further development is needed to implement the partitioning scheme suggested. Appendix F.2 on page 148 contains the updated code.

[46] `https://bugzilla.mozilla.org/show_bug.cgi?id=804303`
[47] `https://code.google.com/p/chromium/issues/detail?id=339604`

3.7 Media Formats

Firefox has numerous media formats supported by the audio and video elements.[48] Currently, Firefox directly supports Ogg (Opus and Vorbis) and Wav audio formats. The AAC and MP3 audio formats are also supported indirectly by relying on support from the operating system or hardware. For video, Firefox supports WebM (VP8 and VP9), and Ogg (Theora). Similar to AAC and MP3, Firefox indirectly supports MP4 (H.264) via OS or hardware support.

iSEC investigated historical bug patterns in these components with an attempt to determine if any are concerning or overwhelmingly unused on the web. Of particular interest are those controlled by five easy-to-change about:config settings, tested on Firefox 29:[49]

1. **media.ogg.enabled** - Disables .OGG-based and .OPUS-based <audio> and .OGV-based <video> elements
2. **media.opus.enabled** - Disables .OPUS-based <audio> elements
3. **media.wave.enabled** - Disables .WAV-based <audio> elements
4. **media.webm.enabled** - Disables .WEBA-based <audio> and .WEBM-based <video> elements
5. **media.apple.mp3.enabled** - Disables .MP3-based <audio> elements (Mac only)

Due to the complexities of audio and video parsing, these components are prone to many bugs, including severe security vulnerabilities. Firefox already has a fairly limited set of supported media formats, however for Tor Browser Bundle it may be best to have media support disabled by default. By requiring users to enable audio or video support on-demand when required by a website, it reduces the risk to these vulnerable formats by limiting unintended processing of potentially malicious audio or video files. Also, as VP9 gains in popularity, VP8 support can be phased out, further reducing attack surface.

[48] https://developer.mozilla.org/en-US/docs/HTML/Supported_media_formats

[49] These settings were tested using http://hpr.dogphilosophy.net/test/, http://www.leanbackplayer.com/test/h5mt.html, and http://www.quirksmode.org/html5/tests/video.html

Historic Security Issues in Media Components

The following table includes only bugs iSEC identified in the media decoders, and do not include bugs occurring in the DOM or JS Cores as a result of the <audio>, <video>, or <canvas> elements.

Title	Impact	Component	Identifier
Use after free reading OGG headers	Critical	OGG	CVE-2011-3005
Heap Buffer Overflow Decoding WAV Data	Critical	WAV Audio	CVE-2012-4186
Potential Memory Corruption When Decoding Ogg Vorbis files	Critical	OGG	CVE-2012-0444
Use After Free in WAV Audio Seeking	Critical	WAV Audio	Bugzilla 821737 (12/2012)
Heap Buffer Overflow in Opus Playback	Critical	OGG	Bugzilla 812847 (11/2012)
Crash in Opus Packet	Critical	OGG	Bugzilla 816994 (11/2012)
Crash in WebMReader	High	OGG	Bugzilla 813562 (11/2012)
Out of bounds read during WAV file decoding	High	WAV Audio	CVE-2014-1497
Crash during WAV audio file decoding	Low	WAV Audio	CVE-2013-1708
Crash during OGG encoding	Low	OGG	Bugzilla 927579 (10/2013)

3.8 Protocol Handlers

iSEC began investigating protocol handlers in Tor Browser Bundle. While the initial concerning protocols, such as `mailto:`, `tel:`, `news://`, and `gopher://` launch external programs or are disabled, some other protocols are also interesting.

In particular, iSEC investigated the `jar:` protocol, which is only supported by Firefox and does not seem to be widely used on the web. This protocol supports URIs of the form `jar:https://example.com/samplearchive.jar!/dir/file.html`, which will open a file contained inside of a zip file. Because large swathes of file types are actually zip files (including .docx, .odt, etc), and that file runs in the context of the hosting domain, there is a possibility for malicious uploads leading to JavaScript execution in the hosting domain's origin.[50] To restrict this, the `network.jar.open-unsafe-types` setting[51] was added[52] and is set to 'false' by default, which does not allow the protocol handler to work unless the MIME type is `application/java-archive` or `application/x-jar` (which in Apache, happens automatically if the filetype is .jar).

iSEC explored the possibility of completely disabling the `jar:` protocol but discovered that, internally, Tor Browser Bundle maps the `app://` protocol implementation to the `jar:` protocol[53] and uses it extensively. iSEC created a patch that defines a setting, `network.jar.block-remote-files` that will prevent Tor Browser Bundle from opening any remote jar files, regardless of MIME type. This patch is included in Appendix D on page 45.

Other protocols of interest that have had security vulnerabilities in the past include `data:`[54] and `view-source://`; however, these are widely used on the web or integral to the functioning of Tor Browser Bundle.

[50] http://www.gnucitizen.org/blog/web-mayhem-firefoxs-jar-protocol-issues/
[51] http://kb.mozillazine.org/Network.jar.open-unsafe-types
[52] https://bugzilla.mozilla.org/show_bug.cgi?id=369814
[53] http://dxr.mozilla.org/mozilla-central/source/netwerk/protocol/app/AppProtocolHandler.cpp
[54] https://bugzilla.mozilla.org/show_bug.cgi?id=255107

3.9 Exposed DOM Objects Enumeration

iSEC identified two ways to enumerate DOM objects exposed by Firefox. These mechanisms will help identify components that should be examined further with a focus on fuzzing, code coverage, privacy, or disabling them entirely.

The first is the WebIDLs specified in `tor-browser/dom/webidl`. These interface definitions represent new DOM components added as a result of W3C specifications – however, iSEC believes not all DOM components exposed are enumerated in WebIDL files.

The DOM test at `dom/tests/mochitest/general/test_interfaces.html` is another location that aims to enumerate all objects in the global namespace. The `dom/bidings/Bindings.conf` file maps these objects to implementations.

More about WebIDLs, DOM object enumeration and bindings can be found at `https://developer.mozilla.org/en-US/docs/Mozilla/WebIDL_bindings`.

3.10 Preference Comparison

iSEC also identified the `modules/libpref/src/init/all.js` file, which appears to contain most preferences set by Firefox and Tor Browser Bundle. iSEC used this file to determine the defaults of preferences as they change between releases. Tor Project could similarly use this file to track changes between ESR releases and attempt to determine if any features have been enabled that may be relevant to the security slider.

3.11 TBB Tests

Using the data from section 3.9, iSEC believes several candidate tests can be created for Tor Browser Bundle. In the short term, these tests are more related to compile-time options, and thus are better suited for the upcoming migration to Firefox ESR 31, along with the preference file explained in section 3.10. The DOM enumeration from section 3.9 can be used to review additional features merged into the browser and review them for privacy concerns. Longer-term, these tests will likely be integral in detecting regressions on the security slider.

iSEC has created a sample test in Appendix B on page 30 that uses the list from `dom/tests/mochitest/general/test_interfaces.html` to enumerate unexpected DOM objects, expected-but-missing DOM objects, and expected-and-seen DOM objects. Note that due to the original `test_interfaces.html` using special post-compilation test harness capabilities (the `SpecialPowers` interface), this list contains a significant number of unexpected and expected-but-missing DOM objects currently.

3.12 browser.fixup.alternate

From a careful reading of the Cure53 SecureDrop Report,[55] iSEC was alerted to to the `browser.fixup.alternate` Firefox settings, which under certain circumstances may automatically append a suffix (such as .com) to URLs. The risk is that the browser attempts to contact a Hidden Service, is unable, and automatically appends .com in an attempt to resolve it.

iSEC investigated the relevant `about:config` settings:

1. `browser.fixup.alternate.suffix` - The suffix, by default ".com", added when a user hits Control+Enter (or on Mac, Meta+Enter) with a single word, to transform "example" into http://www.example.com. This value is also used in conjunction with the prefix in `nsDefaultURIFixup::MakeAlternateURI`, explained below.
2. `browser.fixup.alternate.prefix` - The prefix, by default "www.", used in `nsDefaultURIFixup::MakeAlternateURI` in docshell/base/nsDefaultURIFixup.cpp, which is called by `nsDefaultURIFixup::CreateFixupURI`. The latter function is called in a few places throughout the codebase as documented in Appendix C on page 43 and may lead to information disclosure.
3. `browser.fixup.alternate.enabled` - The preference that controls whether the prefix and suffixed URIs will be tested in `nsDefaultURIFixup::MakeAlternateURI`

Neither Cure53, iSEC, or the Tor Project were able to induce a fixup of a .onion address. However, it is possible that this functionality may change in the future. Because the `browser.fixup.alternate.enabled` preference is only used in a single location to control testing alternate URLs, iSEC recommends that Tor Project investigate disabling this preference, or further asserting that .onion URLs will not be inadvertently leaked if they cannot be contacted.

[55] https://cure53.de/pentest-report_securedrop.pdf

4 Acknowledgments

iSEC would like to thank Mike Perry at the Tor Project for his help determining the scope and providing feedback during this engagement, as well as Open Technology Fund for sponsoring this work.

Additionally, iSEC would like to acknowledge and thank Mozilla, especially Dan Veditz and Mike Hommey, for their assistance and Chris Evans of Google for feedback, suggestions, and his work on browser security in general. Finally, the authors would like to thank consultants internal to iSEC Partners and NCC Group as well as several anonymous friends in the security community for ideas and suggestions offered during conversations.

Appendices

A Bug Classification Glossary

iSEC used the following approximate definitions to guide categorizing bug categories:

- Use After Free (UAF) - A pointer refers to an object that has been freed, and is subsequently dereferenced, leading to use of memory an attacker may control.
- Heap Overwrite - Data is written outside the bounds of the object's allocated heap space
- Heap Overread - Data is read outside the bounds of the object's allocated heap space
- Stack Based Buffer Overwrite - Data is written outside the bounds of the object's allocated stack space
- Memory Leak - Data is disclosed through appropriate buffer bounds, but refers to previously used memory (such as pointers)
- Data Leak - Information about the user's computer, such as local files or screen contents, are exposed.
- Assert - Triggers an assertion in the code
- Use of Uninitialized Memory (UUIM) - Application code uses an uninitialized value, which may be controlled by an attacker
- Type Confusion - Application code interprets an object of one type as another type
- Null Dereference - Application code attempts to dereference a Null pointer
- Double Free - Application Code frees an object twice, possibly corrupting the Heap metadata.

Likewise, iSEC would like to make the following notes about certain components:

- Many of the DOM Core bugs have test cases that use JavaScript to put the DOM in the correct state. It is likely that many of the DOM Core bugs will become unexploitable if JavaScript is disabled.
- In the beginning of classification, iSEC was unfamiliar with the distinction between the general JavaScript Core and the newer Ion JIT engine that can be disabled. Some of the JS Core bugs may belong to the Ion JIT engine.
- In general, this process is imperfect and is designed only to be a rough guide.

B Tor Browser Bundle DOM Tests

```
<html>
<head>
  <title>Tor Browser DOM Test</title>
</head>
<body>

<div style="text-align:center"><h1>TBB DOM Tests</h1></div>

<h2>Unexpected Objects</h2>
<p>These objects were not expected to be present in the Global Namespace. They should
     be carefully examined for security and privacy considerations.</p>
<div id="unexpectedNames"></div>

<h2>Unseen Objects</h2>
<p>These objects were expected to be present in the Global Namespace, but were not.
    They indicate some lack of understanding between how the browser is built and how
     the interfaceNamesInGlobalScope is defined.</p>
<div id="unseenNames"></div>

<h2>Expected Objects</h2>
<p>These objects were expected to be found, and were.</p>
<div style="font-size:smaller" id="seenNames"></div>

<script type="application/javascript">
var objectsIDontCareAbout =
  [
    "interfaceNamesInGlobalScope",
    "objectsIDontCareAbout",
    "Object",
    "Function",
    "eval",
    "window",
    "document",
    "undefined",
    "Boolean",
    "Date",
    "Math",
    "isNaN",
    "isFinite",
    "parseFloat",
    "parseInt",
    "NaN",
    "Infinity",
    "Number",
    "String",
    "escape",
    "unescape",
    "uneval",
```

```
    "decodeURI",
    "encodeURI",
    "decodeURIComponent",
    "encodeURIComponent",
    "Error",
    "InternalError",
    "EvalError",
    "RangeError",
    "ReferenceError",
    "SyntaxError",
    "TypeError",
    "URIError",
    "RegExp",
    "Iterator",
    "StopIteration",
    "Int8Array",
    "Uint8Array",
    "Int16Array",
    "Uint16Array",
    "Int32Array",
    "Uint32Array",
    "Float32Array",
    "Float64Array",
    "Uint8ClampedArray",
    "DataView",
    "ArrayBuffer",
    "WeakMap",
    "Map",
    "Set",
    "Proxy",
    "Image"
  ];
//Taken from Tor Browser's dom/tests/mochitest/general/test_interfaces.html
var interfaceNamesInGlobalScope =
  [
    "AnimationEvent",
    "Array",
    "AsyncScrollEventDetail",
    "Attr",
    "BarProp",
    "BatteryManager",
    "BeforeUnloadEvent",
    "Blob",
    "BlobEvent",
    "BoxObject",
    "CameraCapabilities",
    "CameraControl",
    "CameraManager",
    "CanvasGradient",
    "CanvasPattern",
    "CanvasRenderingContext2D",
```

```
    "CDATASection",
    "CharacterData",
    "ChromeWindow",
    "ClientInformation",
    "ClientRect",
    "ClientRectList",
    "ClipboardEvent",
    "CloseEvent",
    "CommandEvent",
    "Comment",
    "CompositionEvent",
    "Contact",
    "ContactManager",
    "Controllers",
    "Counter",
    "CRMFObject",
    "Crypto",
    "CryptoDialogs",
    "CSS2Properties",
    "CSSCharsetRule",
    "CSSConditionRule",
    "CSSFontFaceRule",
    "CSSFontFeatureValuesRule",
    "CSSGroupingRule",
    "CSSImportRule",
    "CSSMediaRule",
    "CSSMozDocumentRule",
    "CSSPageRule",
    "CSSPrimitiveValue",
    "CSSRule",
    "CSSRuleList",
    "CSSStyleDeclaration",
    "CSSStyleRule",
    "CSSStyleSheet",
    "CSSSupportsRule",
    "CSSUnknownRule",
    "CSSValue",
    "CSSValueList",
    "CustomEvent",
    "DataChannel",
    "DataContainerEvent",
    "DataErrorEvent",
    "DataTransfer",
    "DesktopNotification",
    "DesktopNotificationCenter",
    "DeviceAcceleration",
    "DeviceLightEvent",
    "DeviceMotionEvent",
    "DeviceOrientationEvent",
    "DeviceProximityEvent",
    "DeviceRotationRate",
```

```
148     "DeviceStorage",
149     "DeviceStorageChangeEvent",
150     "DeviceStorageCursor",
151     "Document",
152     "DocumentFragment",
153     "DocumentTouch",
154     "DocumentType",
155     "DocumentXBL",
156     "DOMCursor",
157     "DOMError",
158     "DOMException",
159     "DOMImplementation",
160     "DOMRequest",
161     "DOMSettableTokenList",
162     "DOMStringList",
163     "DOMStringMap",
164     "DOMTokenList",
165     "DOMTransactionEvent",
166     "DragEvent",
167     "Element",
168     "ElementCSSInlineStyle",
169     "ElementReplaceEvent",
170     "ElementTimeControl",
171     "Event",
172     "EventListener",
173     "EventListenerInfo",
174     "EventSource",
175     "EventTarget",
176     "File",
177     "FileHandle",
178     "FileList",
179     "FileReader",
180     "FileRequest",
181     "FocusEvent",
182     "FontFace",
183     "FontFaceList",
184     "FormData",
185     "Gamepad",
186     "GamepadAxisMoveEvent",
187     "GamepadButtonEvent",
188     "GamepadEvent",
189     "GeoGeolocation",
190     "GeoPosition",
191     "GeoPositionCallback",
192     "GeoPositionCoords",
193     "GeoPositionError",
194     "GeoPositionErrorCallback",
195     "GetUserMediaErrorCallback",
196     "GetUserMediaSuccessCallback",
197     "GlobalObjectConstructor",
198     "GlobalPropertyInitializer",
```

```
    "HashChangeEvent",
    "History",
    "HTMLAnchorElement",
    "HTMLAppletElement",
    "HTMLAreaElement",
    "HTMLAudioElement",
    "HTMLBaseElement",
    "HTMLBodyElement",
    "HTMLBRElement",
    "HTMLButtonElement",
    "HTMLByteRanges",
    "HTMLCanvasElement",
    "HTMLCollection",
    "HTMLCommandElement",
    "HTMLDataListElement",
    "HTMLDirectoryElement",
    "HTMLDivElement",
    "HTMLDListElement",
    "HTMLDocument",
    "HTMLElement",
    "HTMLEmbedElement",
    "HTMLFieldSetElement",
    "HTMLFontElement",
    "HTMLFormElement",
    "HTMLFrameElement",
    "HTMLFrameSetElement",
    "HTMLHeadElement",
    "HTMLHeadingElement",
    "HTMLHRElement",
    "HTMLHtmlElement",
    "HTMLIFrameElement",
    "HTMLImageElement",
    "HTMLInputElement",
    "HTMLLabelElement",
    "HTMLLegendElement",
    "HTMLLIElement",
    "HTMLLinkElement",
    "HTMLMapElement",
    "HTMLMediaElement",
    "HTMLMenuElement",
    "HTMLMenuItemElement",
    "HTMLMetaElement",
    "HTMLMeterElement",
    "HTMLModElement",
    "HTMLObjectElement",
    "HTMLOListElement",
    "HTMLOptGroupElement",
    "HTMLOptionElement",
    "HTMLOptionsCollection",
    "HTMLOutputElement",
    "HTMLParagraphElement",
```

```
    "HTMLParamElement",
    "HTMLPreElement",
    "HTMLProgressElement",
    "HTMLPropertiesCollection",
    "HTMLQuoteElement",
    "HTMLScriptElement",
    "HTMLSelectElement",
    "HTMLSourceElement",
    "HTMLStyleElement",
    "HTMLTableCaptionElement",
    "HTMLTableCellElement",
    "HTMLTableColElement",
    "HTMLTableElement",
    "HTMLTableRowElement",
    "HTMLTableSectionElement",
    "HTMLTextAreaElement",
    "HTMLTitleElement",
    "HTMLUListElement",
    "HTMLUnknownElement",
    "HTMLVideoElement",
    "IDBCursor",
    "IDBCursorWithValue",
    "IDBDatabase",
    "IDBFactory",
    "IDBIndex",
    "IDBKeyRange",
    "IDBObjectStore",
    "IDBOpenDBRequest",
    "IDBRequest",
    "IDBTransaction",
    "IDBVersionChangeEvent",
    "ImageData",
    "ImageDocument",
    "JSON",
    "JSWindow",
    "KeyEvent",
    "LinkStyle",
    "LoadStatus",
    "LocalMediaStream",
    "Location",
    "LockedFile",
    "LSProgressEvent",
    "MediaError",
    "MediaList",
    "MediaQueryList",
    "MediaQueryListListener",
    "MediaStream",
    "MessageEvent",
    "MimeType",
    "MimeTypeArray",
    "ModalContentWindow",
```

```
    "MouseEvent",
    "MouseScrollEvent",
    "MozAlarmsManager",
    "MozApplicationEvent",
    "MozBlobBuilder",
    "MozBrowserFrame",
    "MozCanvasPrintState",
    "MozConnection",
    "MozContactChangeEvent",
    "MozCSSKeyframeRule",
    "MozCSSKeyframesRule",
    "MozMmsEvent",
    "MozMmsMessage",
    "MozMobileCellInfo",
    "MozMobileConnectionInfo",
    "MozMobileMessageManager",
    "MozMobileMessageThread",
    "MozMobileNetworkInfo",
    "MozNamedAttrMap",
    "MozNavigatorMobileMessage",
    "MozNavigatorNetwork",
    "MozNavigatorSms",
    "MozNavigatorTime",
    "MozNetworkStats",
    "MozNetworkStatsData",
    "MozNetworkStatsManager",
    "MozPowerManager",
    "MozSettingsEvent",
    "MozSmsEvent",
    "MozSmsFilter",
    "MozSmsManager",
    "MozSmsMessage",
    "MozSmsSegmentInfo",
    "MozTimeManager",
    "MozTouchEvent",
    "MozWakeLock",
    "MozWakeLockListener",
    "MutationEvent",
    "MutationObserver",
    "MutationRecord",
    "Navigator",
    "NavigatorCamera",
    "NavigatorDesktopNotification",
    "NavigatorDeviceStorage",
    "NavigatorGeolocation",
    "NavigatorUserMedia",
    "Node",
    "NodeFilter",
    "NodeIterator",
    "NodeList",
    "NodeSelector",
```

```
352     "NotifyAudioAvailableEvent",
353     "NotifyPaintEvent",
354     "NSEditableElement",
355     "NSEvent",
356     "NSRGBAColor",
357     "NSXPathExpression",
358     "OfflineResourceList",
359     "OpenWindowEventDetail",
360     "PageTransitionEvent",
361     "PaintRequest",
362     "PaintRequestList",
363     "Parser",
364     "ParserJS",
365     "PaymentRequestInfo",
366     "Performance",
367     "PerformanceNavigation",
368     "PerformanceTiming",
369     "PermissionSettings",
370     "Pkcs11",
371     "Plugin",
372     "PluginArray",
373     "PopStateEvent",
374     "PopupBlockedEvent",
375     "ProcessingInstruction",
376     "ProgressEvent",
377     "PropertyNodeList",
378     "PushManager",
379     "Range",
380     "Rect",
381     "RequestService",
382     "RGBColor",
383     "RTCIceCandidate",
384     "RTCPeerConnection",
385     "RTCSessionDescription",
386     "Screen",
387     "ScrollAreaEvent",
388     "Selection",
389     "Serializer",
390     "SettingsLock",
391     "SettingsManager",
392     "SimpleGestureEvent",
393     "SmartCardEvent",
394     "SpeechRecognitionError",
395     "SpeechRecognitionEvent",
396     "SpeechSynthesisEvent",
397     "Storage",
398     "StorageEvent",
399     "StorageIndexedDB",
400     "StorageItem",
401     "StorageManager",
402     "StorageObsolete",
```

```
403     "StyleRuleChangeEvent",
404     "StyleSheet",
405     "StyleSheetApplicableStateChangeEvent",
406     "StyleSheetChangeEvent",
407     "StyleSheetList",
408     "SVGAElement",
409     "SVGAltGlyphElement",
410     "SVGAngle",
411     "SVGAnimatedAngle",
412     "SVGAnimatedBoolean",
413     "SVGAnimatedEnumeration",
414     "SVGAnimatedInteger",
415     "SVGAnimatedLength",
416     "SVGAnimatedLengthList",
417     "SVGAnimatedNumber",
418     "SVGAnimatedNumberList",
419     "SVGAnimatedPathData",
420     "SVGAnimatedPoints",
421     "SVGAnimatedPreserveAspectRatio",
422     "SVGAnimatedRect",
423     "SVGAnimatedString",
424     "SVGAnimatedTransformList",
425     "SVGAnimateElement",
426     "SVGAnimateMotionElement",
427     "SVGAnimateTransformElement",
428     "SVGAnimationElement",
429     "SVGCircleElement",
430     "SVGClipPathElement",
431     "SVGComponentTransferFunctionElement",
432     "SVGDefsElement",
433     "SVGDescElement",
434     "SVGDocument",
435     "SVGElement",
436     "SVGEllipseElement",
437     "SVGEvent",
438     "SVGFEBlendElement",
439     "SVGFEColorMatrixElement",
440     "SVGFEComponentTransferElement",
441     "SVGFECompositeElement",
442     "SVGFEConvolveMatrixElement",
443     "SVGFEDiffuseLightingElement",
444     "SVGFEDisplacementMapElement",
445     "SVGFEDistantLightElement",
446     "SVGFEFloodElement",
447     "SVGFEFuncAElement",
448     "SVGFEFuncBElement",
449     "SVGFEFuncGElement",
450     "SVGFEFuncRElement",
451     "SVGFEGaussianBlurElement",
452     "SVGFEImageElement",
453     "SVGFEMergeElement",
```

```
454     "SVGFEMergeNodeElement",
455     "SVGFEMorphologyElement",
456     "SVGFEOffsetElement",
457     "SVGFEPointLightElement",
458     "SVGFESpecularLightingElement",
459     "SVGFESpotLightElement",
460     "SVGFETileElement",
461     "SVGFETurbulenceElement",
462     "SVGFilterElement",
463     "SVGFilterPrimitiveStandardAttributes",
464     "SVGFitToViewBox",
465     "SVGForeignObjectElement",
466     "SVGGElement",
467     "SVGGradientElement",
468     "SVGImageElement",
469     "SVGLength",
470     "SVGLengthList",
471     "SVGLinearGradientElement",
472     "SVGLineElement",
473     "SVGLocatable",
474     "SVGMarkerElement",
475     "SVGMaskElement",
476     "SVGMatrix",
477     "SVGMetadataElement",
478     "SVGMpathElement",
479     "SVGNumber",
480     "SVGNumberList",
481     "SVGPathElement",
482     "SVGPathSeg",
483     "SVGPathSegArcAbs",
484     "SVGPathSegArcRel",
485     "SVGPathSegClosePath",
486     "SVGPathSegCurvetoCubicAbs",
487     "SVGPathSegCurvetoCubicRel",
488     "SVGPathSegCurvetoCubicSmoothAbs",
489     "SVGPathSegCurvetoCubicSmoothRel",
490     "SVGPathSegCurvetoQuadraticAbs",
491     "SVGPathSegCurvetoQuadraticRel",
492     "SVGPathSegCurvetoQuadraticSmoothAbs",
493     "SVGPathSegCurvetoQuadraticSmoothRel",
494     "SVGPathSegLinetoAbs",
495     "SVGPathSegLinetoHorizontalAbs",
496     "SVGPathSegLinetoHorizontalRel",
497     "SVGPathSegLinetoRel",
498     "SVGPathSegLinetoVerticalAbs",
499     "SVGPathSegLinetoVerticalRel",
500     "SVGPathSegList",
501     "SVGPathSegMovetoAbs",
502     "SVGPathSegMovetoRel",
503     "SVGPatternElement",
504     "SVGPoint",
```

```
    "SVGPointList",
    "SVGPolygonElement",
    "SVGPolylineElement",
    "SVGPreserveAspectRatio",
    "SVGRadialGradientElement",
    "SVGRect",
    "SVGRectElement",
    "SVGScriptElement",
    "SVGSetElement",
    "SVGStopElement",
    "SVGStringList",
    "SVGStylable",
    "SVGStyleElement",
    "SVGSVGElement",
    "SVGSwitchElement",
    "SVGSymbolElement",
    "SVGTests",
    "SVGTextContentElement",
    "SVGTextElement",
    "SVGTextPathElement",
    "SVGTextPositioningElement",
    "SVGTitleElement",
    "SVGTransform",
    "SVGTransformable",
    "SVGTransformList",
    "SVGTSpanElement",
    "SVGUnitTypes",
    "SVGURIReference",
    "SVGUseElement",
    "SVGViewElement",
    "SVGViewSpec",
    "SVGZoomAndPan",
    "SVGZoomEvent",
    "TCPSocket",
    "Text",
    "TextMetrics",
    "TimeEvent",
    "TimeRanges",
    "ToString",
    "Touch",
    "TouchEvent",
    "TouchList",
    "TransitionEvent",
    "TreeColumn",
    "TreeColumns",
    "TreeContentView",
    "TreeSelection",
    "TreeWalker",
    "UIEvent",
    "UndoManager",
    "URL",
```

```
    "UserDataHandler",
    "UserProximityEvent",
    "USSDReceivedEvent",
    "ValidityState",
    "WebGLRenderingContext",
    "WebSocket",
    "WheelEvent",
    "Window",
    "WindowCollection",
    "WindowInternal",
    "WindowPerformance",
    "WindowUtils",
    "XMLDocument",
    "XMLHttpRequest",
    "XMLHttpRequestEventTarget",
    "XMLHttpRequestUpload",
    "XPathEvaluator",
    "XPathExpression",
    "XPathNamespace",
    "XPathNSResolver",
    "XPathResult",
    "XSLTProcessor",
    "XULButtonElement",
    "XULCheckboxElement",
    "XULCommandDispatcher",
    "XULCommandEvent",
    "XULContainerElement",
    "XULContainerItemElement",
    "XULControlElement",
    "XULDescriptionElement",
    "XULDocument",
    "XULElement",
    "XULImageElement",
    "XULLabeledControlElement",
    "XULLabelElement",
    "XULMenuListElement",
    "XULMultiSelectControlElement",
    "XULPopupElement",
    "XULRelatedElement",
    "XULSelectControlElement",
    "XULSelectControlItemElement",
    "XULTemplateBuilder",
    "XULTextBoxElement",
    "XULTreeBuilder",
    "XULTreeElement",
  ]

var populateLists = function() {
  var seenList = [];
  var unseenList = [];
  var unexpectedList = [];
```

```
  var allObjects = Object.getOwnPropertyNames(window);
  for(var i in allObjects) {
      name = allObjects[i];
      if(interfaceNamesInGlobalScope.indexOf(name) >= 0 ||
         objectsIDontCareAbout.indexOf(name) >= 0){
        seenList.push(name);
      }
      else {
        unexpectedList.push(name);
      }
  }
  for(var i in interfaceNamesInGlobalScope) {
    name = interfaceNamesInGlobalScope[i];
    if(allObjects.indexOf(name) < 0) {
      unseenList.push(name);
    }
  }

  unseenNames = '<ol>';
  for(var i in unseenList) {
    unseenNames += '<li>' + unseenList[i] + '</li>\n';
  }
  unseenNames += '</ol>';

  seenNames = '<ol>';
  for(var i in seenList) {
    seenNames += '<li>' + seenList[i] + '</li>\n';
  }
  seenNames += '</ol>';

  unexpectedNames = '<ol>';
  for(var i in unexpectedList) {
    unexpectedNames += '<li>' + unexpectedList[i] + '</li>\n';
  }
  unexpectedNames += '</ol>';

  document.getElementById('unseenNames').innerHTML = unseenNames;
  document.getElementById('seenNames').innerHTML = seenNames;
  document.getElementById('unexpectedNames').innerHTML = unexpectedNames;
}
setTimeout(populateLists, 1000);
</script>

</body>
</html>
```

Listing 1: Enumerating DOM Objects

C CreateFixupURL Calls

`nsDefaultURIFixup::CreateFixupURI` will only use the `browser.fixup.alternate.suffix` value to create a new URI if the flag FIXUP_FLAGS_MAKE_ALTERNATE_URI is provided. Searching for this flag yields the following two results:

```
NS_IMETHODIMP
nsScriptSecurityManager::CheckLoadURIStrWithPrincipal(nsIPrincipal* aPrincipal,
                    const nsACString& aTargetURIStr, uint32_t aFlags) {
    nsresult rv;
    nsCOMPtr<nsIURI> target;
    rv = NS_NewURI(getter_AddRefs(target), aTargetURIStr,
                   nullptr, nullptr, sIOService);
    NS_ENSURE_SUCCESS(rv, rv);

    rv = CheckLoadURIWithPrincipal(aPrincipal, target, aFlags);
    NS_ENSURE_SUCCESS(rv, rv);

    // Now start testing fixup -- since aTargetURIStr is a string, not
    // an nsIURI, we may well end up fixing it up before loading.
    // Note: This needs to stay in sync with the nsIURIFixup api.
    nsCOMPtr<nsIURIFixup> fixup = do_GetService(NS_URIFIXUP_CONTRACTID);
    if (!fixup) {
        return rv;
    }

    uint32_t flags[] = {
        nsIURIFixup::FIXUP_FLAG_NONE,
        nsIURIFixup::FIXUP_FLAG_ALLOW_KEYWORD_LOOKUP,
        nsIURIFixup::FIXUP_FLAGS_MAKE_ALTERNATE_URI,
        nsIURIFixup::FIXUP_FLAG_ALLOW_KEYWORD_LOOKUP |
        nsIURIFixup::FIXUP_FLAGS_MAKE_ALTERNATE_URI
    };

    for (uint32_t i = 0; i < ArrayLength(flags); ++i) {
        rv = fixup->CreateFixupURI(aTargetURIStr, flags[i], nullptr,
                                   getter_AddRefs(target));
        NS_ENSURE_SUCCESS(rv, rv);

        rv = CheckLoadURIWithPrincipal(aPrincipal, target, aFlags);
        NS_ENSURE_SUCCESS(rv, rv);
    }

    return rv;
}
```

Listing 2: caps/src/nsScriptSecurityManager.cpp

```
// Edited slightly for brevity
// Now try change the address, e.g. turn http://foo into
// http://www.foo.com
if (aStatus == NS_ERROR_UNKNOWN_HOST ||
    aStatus == NS_ERROR_NET_RESET) {
    bool doCreateAlternate = true;

    // Skip fixup for anything except a normal document load
    // operation on the topframe.
    if (mLoadType != LOAD_NORMAL || !isTopFrame)
        doCreateAlternate = false;
    else {
        // Test if keyword lookup produced a new URI or not
        if (newURI) {
            bool sameURI = false;
            url->Equals(newURI, &sameURI);
            if (!sameURI) {
                // Keyword lookup made a new URI so no need to try
                // an alternate one.
                doCreateAlternate = false;
            }
        }
    }
    if (doCreateAlternate) {
        newURI = nullptr;
        newPostData = nullptr;
        sURIFixup->CreateFixupURI(oldSpec,
          nsIURIFixup::FIXUP_FLAGS_MAKE_ALTERNATE_URI,
               getter_AddRefs(newPostData), getter_AddRefs(newURI));
    }
}

// Did we make a new URI that is different to the old one? If so
// load it.
if (newURI) {
    // Make sure the new URI is different from the old one,
    // otherwise there's little point trying to load it again.
    bool sameURI = false;
    url->Equals(newURI, &sameURI);
    if (!sameURI) {
        nsAutoCString newSpec;
        newURI->GetSpec(newSpec);
        NS_ConvertUTF8toUTF16 newSpecW(newSpec);

        return LoadURI(newSpecW.get(),
               LOAD_FLAGS_NONE, nullptr, newPostData, nullptr);
    } }
```

Listing 3: docshell/base/nsDocShell.cpp

D Configuration Setting to Block All Remote JAR Files

```
From 1fc4163cfae73f7de62c644718204f644c11db41 Mon Sep 17 00:00:00 2001
From: Jeff Gibat <jgibat@isecpartners.com>
Date: Wed, 21 May 2014 20:23:32 +0000
Subject: [PATCH] adding a config preference that allows a user to block all
 remote jar files regardless of content type

---
 modules/libjar/nsJARChannel.cpp |    6 ++++++
 modules/libpref/src/init/all.js |    3 +++
 2 files changed, 9 insertions(+)

diff --git a/modules/libjar/nsJARChannel.cpp b/modules/libjar/nsJARChannel.cpp
index 22b483a..47a212e 100644
--- a/modules/libjar/nsJARChannel.cpp
+++ b/modules/libjar/nsJARChannel.cpp
@@ -902,6 +902,12 @@ nsJARChannel::OnDownloadComplete(nsIDownloader *downloader,
         mContentDisposition = NS_GetContentDispositionFromHeader(
             mContentDispositionHeader, this);
     }

+    // here we check preferences to see if all remote jar support should be disabled
+    if (Preferences::GetBool("network.jar.block-remote-files", true)) {
+        mIsUnsafe = true;
+        status = NS_ERROR_UNSAFE_CONTENT_TYPE;
+    }
+
     if (NS_SUCCEEDED(status) && mIsUnsafe &&
         !Preferences::GetBool("network.jar.open-unsafe-types", false)) {
         status = NS_ERROR_UNSAFE_CONTENT_TYPE;
diff --git a/modules/libpref/src/init/all.js b/modules/libpref/src/init/all.js
index 0a2588d..3623e38 100644
--- a/modules/libpref/src/init/all.js
+++ b/modules/libpref/src/init/all.js
@@ -1107,6 +1107,9 @@ pref("dom.server-events.default-reconnection-time", 5000); // in milliseconds
 // by the jar channel.
 pref("network.jar.open-unsafe-types", false);

+// If true, remote JAR files will not be opened, regardless of content type
+pref("network.jar.block-remote-files", true);
+
 // This preference, if true, causes all UTF-8 domain names to be normalized to
 // punycode.  The intention is to allow UTF-8 domain names as input, but never
 // generate them from punycode.
--
```

```
44 1.7.9.5
```

Listing 4: Sample Patch For Blocking All Remote JAR Files

E Enable Assertions Patches

E.1 System Assertions

```
diff --git a/db/sqlite3/src/sqlite3.c b/db/sqlite3/src/sqlite3.c
index deef460..c633695 100644
--- a/db/sqlite3/src/sqlite3.c
+++ b/db/sqlite3/src/sqlite3.c
@@ -8083,7 +8083,7 @@ SQLITE_PRIVATE void sqlite3HashClear(Hash*);
 #include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
-#include <assert.h>
+#include <assert-orig.h>
 #include <stddef.h>

 /*
diff --git a/media/libnestegg/src/halloc.c b/media/libnestegg/src/halloc.c
index 5758fc0..5382c56 100644
--- a/media/libnestegg/src/halloc.c
+++ b/media/libnestegg/src/halloc.c
@@ -24,7 +24,7 @@
  */
 typedef struct hblock
 {
-#ifndef NDEBUG
+#ifndef TOR_NASSERT
 #define HH_MAGIC    0x20040518L
  long          magic;
 #endif
diff --git a/toolkit/crashreporter/google-breakpad/src/common/dwarf/dwarf2reader.cc b
    /toolkit/crashreporter/google-breakpad/src/common/dwarf/dwarf2reader.cc
index 7d0b8af..4076ea8 100644
--- a/toolkit/crashreporter/google-breakpad/src/common/dwarf/dwarf2reader.cc
+++ b/toolkit/crashreporter/google-breakpad/src/common/dwarf/dwarf2reader.cc
@@ -86,7 +86,7 @@ void CompilationUnit::ReadAbbrevs() {
   const char* abbrev_start = iter->second.first +
                                       header_.abbrev_offset;
   const char* abbrevptr = abbrev_start;
-#ifndef NDEBUG
+#ifndef TOR_NASSERT
   const uint64 abbrev_length = iter->second.second - header_.abbrev_offset;
 #endif
```

Listing 5: Sample Patch For Enabling Standard System Assertions From assert.h

```
--- /usr/include/assert-orig.h  2014-05-05 22:17:11.711269515 +0000
+++ /usr/include/assert.h 2014-05-05 22:08:43.683270829 +0000
@@ -47,7 +47,7 @@
    If NDEBUG is defined, do nothing.
    If not, and EXPRESSION is zero, print an error message and abort.  */

-#ifdef NDEBUG
+#ifdef TOR_NASSERT /* NDEBUG */

 # define assert(expr)    (__ASSERT_VOID_CAST (0))
```

Listing 6: Sample Patch For Enabling Standard System Assertions From assert.h

E.2 nsCOMPtr Assertions

```
diff --git a/xpcom/glue/Makefile.in b/xpcom/glue/Makefile.in
index f41ac6d..07242f8 100644
--- a/xpcom/glue/Makefile.in
+++ b/xpcom/glue/Makefile.in
@@ -33,6 +33,7 @@ SDK_HEADERS = \
    nsCycleCollectorUtils.h \
    nsDataHashtable.h \
    nsDebug.h \
+   nsDebugTor.h \
    nsDeque.h \
    nsEnumeratorUtils.h \
    nsHashKeys.h \
diff --git a/xpcom/glue/nsCOMPtr.h b/xpcom/glue/nsCOMPtr.h
index d082928..66ccf4a 100644
--- a/xpcom/glue/nsCOMPtr.h
+++ b/xpcom/glue/nsCOMPtr.h
@@ -25,9 +25,9 @@
 #include "mozilla/NullPtr.h"

   // Wrapping includes can speed up compiles (see "Large Scale C++ Software Design")
-#ifndef nsDebug_h___
-#include "nsDebug.h"
-  // for |NS_ABORT_IF_FALSE|, |NS_ASSERTION|
+#ifndef nsDebugTor_h___
+#include "nsDebugTor.h"
+  // for |TBB_NS_ABORT_IF_FALSE|, |TBB_NS_ASSERTION|
 #endif

 #ifndef nsISupportsUtils_h__
@@ -542,7 +542,7 @@ class nsCOMPtr MOZ_FINAL
            if ( mRawPtr )
              {
                nsCOMPtr<T> query_result( do_QueryInterface(mRawPtr) );
-               NS_ASSERTION(query_result.get() == mRawPtr, "QueryInterface needed");
+               TBB_NS_ASSERTION(query_result.get() == mRawPtr, "QueryInterface needed
    ");
              }
         }

@@ -804,7 +804,7 @@ class nsCOMPtr MOZ_FINAL
           // parameters where rhs bay be a T** or an I** where I is a base class
           // of T.
         {
-          NS_ASSERTION(rhs, "Null pointer passed to forget!");
+          TBB_NS_ASSERTION(rhs, "Null pointer passed to forget!");
           NSCAP_LOG_RELEASE(this, mRawPtr);
           *rhs = get();
           mRawPtr = 0;
```

```
@@ -836,7 +836,7 @@ class nsCOMPtr MOZ_FINAL
       T*
       operator->() const
         {
-          NS_ABORT_IF_FALSE(mRawPtr != 0, "You can't dereference a NULL nsCOMPtr with operator->().");
+          TBB_NS_ABORT_IF_FALSE(mRawPtr != 0, "You can't dereference a NULL nsCOMPtr with operator->().");
           return get();
         }

@@ -860,7 +860,7 @@ class nsCOMPtr MOZ_FINAL
       T&
       operator*() const
         {
-          NS_ABORT_IF_FALSE(mRawPtr != 0, "You can't dereference a NULL nsCOMPtr with operator*().");
+          TBB_NS_ABORT_IF_FALSE(mRawPtr != 0, "You can't dereference a NULL nsCOMPtr with operator*().");
           return *get();
         }

@@ -1109,7 +1109,7 @@ class nsCOMPtr<nsISupports>
           // Useful to avoid unnecessary AddRef/Release pairs with "out"
           // parameters.
         {
-          NS_ASSERTION(rhs, "Null pointer passed to forget!");
+          TBB_NS_ASSERTION(rhs, "Null pointer passed to forget!");
           *rhs = 0;
           swap(*rhs);
         }
@@ -1143,7 +1143,7 @@ class nsCOMPtr<nsISupports>
       nsISupports*
       operator->() const
         {
-          NS_ABORT_IF_FALSE(mRawPtr != 0, "You can't dereference a NULL nsCOMPtr with operator->().");
+          TBB_NS_ABORT_IF_FALSE(mRawPtr != 0, "You can't dereference a NULL nsCOMPtr with operator->().");
           return get();
         }

@@ -1168,7 +1168,7 @@ class nsCOMPtr<nsISupports>
       nsISupports&
       operator*() const
         {
-          NS_ABORT_IF_FALSE(mRawPtr != 0, "You can't dereference a NULL nsCOMPtr with operator*().");
+          TBB_NS_ABORT_IF_FALSE(mRawPtr != 0, "You can't dereference a NULL nsCOMPtr with operator*().");
           return *get();
```

```
91          }
92
93  diff --git a/xpcom/glue/nsDebugTor.h b/xpcom/glue/nsDebugTor.h
94  new file mode 100644
95  index 0000000..343e84e
96  --- /dev/null
97  +++ b/xpcom/glue/nsDebugTor.h
98  @@ -0,0 +1,371 @@
99  +/* -*- Mode: C++; tab-width: 4; indent-tabs-mode: nil; c-basic-offset: 2 -*- */
100 +/* This Source Code Form is subject to the terms of the Mozilla Public
101 + * License, v. 2.0. If a copy of the MPL was not distributed with this
102 + * file, You can obtain one at http://mozilla.org/MPL/2.0/. */
103 +
104 +#ifndef nsDebugTor_h___
105 +#define nsDebugTor_h___
106 +
107 +#ifndef nscore_h___
108 +#include "nscore.h"
109 +#endif
110 +
111 +#ifndef nsError_h__
112 +#include "nsError.h"
113 +#endif
114 +
115 +#include "nsXPCOM.h"
116 +#include "mozilla/Assertions.h"
117 +#include "mozilla/Likely.h"
118 +
119 +#ifndef TOR_NASSERT
120 +#include "prprf.h"
121 +#endif
122 +
123 +#ifndef TOR_NASSERT
124 +
125 +/**
126 + * Abort the execution of the program if the expression evaluates to
127 + * false.
128 + *
129 + * There is no status value returned from the macro.
130 + *
131 + * Note that the non-debug version of this macro does <b>not</b>
132 + * evaluate the expression argument. Hence side effect statements
133 + * as arguments to the macro will yield improper execution in a
134 + * non-debug build. For example:
135 + *
136 + *      TBB_NS_ABORT_IF_FALSE(0 == foo++, "yikes foo should be zero");
137 + *
138 + * Note also that the non-debug version of this macro does <b>not</b>
139 + * evaluate the message argument.
140 + */
141 +#define TBB_NS_ABORT_IF_FALSE(_expr, _msg)                       \
```

```
+  do {                                                        \
+    if (!(_expr)) {                                           \
+      NS_DebugBreak(NS_DEBUG_ABORT, _msg, #_expr, __FILE__, __LINE__); \
+    }                                                         \
+  } while(0)
+
+/**
+ * Warn if a given condition is false.
+ *
+ * Program execution continues past the usage of this macro.
+ *
+ * Note also that the non-debug version of this macro does <b>not</b>
+ * evaluate the message argument.
+ */
+#define TBB_NS_WARN_IF_FALSE(_expr,_msg)                        \
+  do {                                                        \
+    if (!(_expr)) {                                           \
+      NS_DebugBreak(TBB_NS_DEBUG_WARNING, _msg, #_expr, __FILE__, __LINE__); \
+    }                                                         \
+  } while(0)
+
+/**
+ * Test a precondition for truth. If the expression is not true then
+ * trigger a program failure.
+ */
+#define TBB_NS_PRECONDITION(expr, str)                          \
+  do {                                                        \
+    if (!(expr)) {                                            \
+      NS_DebugBreak(NS_DEBUG_ASSERTION, str, #expr, __FILE__, __LINE__); \
+    }                                                         \
+  } while(0)
+
+/**
+ * Test an assertion for truth. If the expression is not true then
+ * trigger a program failure.
+ */
+#define TBB_NS_ASSERTION(expr, str)                             \
+  do {                                                        \
+    if (!(expr)) {                                            \
+      NS_DebugBreak(NS_DEBUG_ASSERTION, str, #expr, __FILE__, __LINE__); \
+    }                                                         \
+  } while(0)
+
+/**
+ * Test a post-condition for truth. If the expression is not true then
+ * trigger a program failure.
+ */
+#define TBB_NS_POSTCONDITION(expr, str)                         \
+  do {                                                        \
+    if (!(expr)) {                                            \
+      NS_DebugBreak(NS_DEBUG_ASSERTION, str, #expr, __FILE__, __LINE__); \
```

```
+    }                                                          \
+  } while(0)
+
+/**
+ * This macros triggers a program failure if executed. It indicates that
+ * an attempt was made to execute some unimplemented functionality.
+ */
+#define TBB_NS_NOTYETIMPLEMENTED(str)                          \
+  NS_DebugBreak(NS_DEBUG_ASSERTION, str, "NotYetImplemented", __FILE__, __LINE__)
+
+/**
+ * This macros triggers a program failure if executed. It indicates that
+ * an attempt was made to execute some unimplemented functionality.
+ */
+#define TBB_NS_NOTREACHED(str)                                 \
+  NS_DebugBreak(NS_DEBUG_ASSERTION, str, "Not Reached", __FILE__, __LINE__)
+
+/**
+ * Log an error message.
+ */
+#define TBB_NS_ERROR(str)                                      \
+  NS_DebugBreak(NS_DEBUG_ASSERTION, str, "Error", __FILE__, __LINE__)
+
+/**
+ * Log a warning message.
+ */
+#define TBB_NS_WARNING(str)                                    \
+  NS_DebugBreak(TBB_NS_DEBUG_WARNING, str, nullptr, __FILE__, __LINE__)
+
+/**
+ * Trigger an abort
+ */
+#define TBB_NS_ABORT()                                         \
+  NS_DebugBreak(NS_DEBUG_ABORT, nullptr, nullptr, __FILE__, __LINE__)
+
+/**
+ * Cause a break
+ */
+#define TBB_NS_BREAK()                                         \
+  NS_DebugBreak(TBB_NS_DEBUG_BREAK, nullptr, nullptr, __FILE__, __LINE__)
+
+#else /* DEBUG */
+
+/**
+ * The non-debug version of these macros do not evaluate the
+ * expression or the message arguments to the macro.
+ */
+#define TBB_NS_ABORT_IF_FALSE(_expr, _msg) do { /* nothing */ } while(0)
+#define TBB_NS_WARN_IF_FALSE(_expr, _msg)  do { /* nothing */ } while(0)
+#define TBB_NS_PRECONDITION(expr, str)     do { /* nothing */ } while(0)
+#define TBB_NS_ASSERTION(expr, str)        do { /* nothing */ } while(0)
```

```
+#define TBB_NS_POSTCONDITION(expr, str)    do { /* nothing */ } while(0)
+#define TBB_NS_NOTYETIMPLEMENTED(str)      do { /* nothing */ } while(0)
+#define TBB_NS_NOTREACHED(str)             do { /* nothing */ } while(0)
+#define TBB_NS_ERROR(str)                  do { /* nothing */ } while(0)
+#define TBB_NS_WARNING(str)                do { /* nothing */ } while(0)
+#define TBB_NS_ABORT()                     do { /* nothing */ } while(0)
+#define TBB_NS_BREAK()                     do { /* nothing */ } while(0)
+
+#endif /* TOR_ASSERT */
+
+/*****************************************************************************
+** Macros for static assertions.  These are used by the sixgill tool.
+** When the tool is not running these macros are no-ops.
+*****************************************************************************/
+
+/* Avoid name collision if included with other headers defining annotations. */
+#ifndef HAVE_STATIC_ANNOTATIONS
+#define HAVE_STATIC_ANNOTATIONS
+
+#ifdef XGILL_PLUGIN
+
+#define STATIC_PRECONDITION(COND)         __attribute__((precondition(#COND)))
+#define STATIC_PRECONDITION_ASSUME(COND)  __attribute__((precondition_assume(#COND))
    )
+#define STATIC_POSTCONDITION(COND)        __attribute__((postcondition(#COND)))
+#define STATIC_POSTCONDITION_ASSUME(COND) __attribute__((postcondition_assume(#COND)
    ))
+#define STATIC_INVARIANT(COND)            __attribute__((invariant(#COND)))
+#define STATIC_INVARIANT_ASSUME(COND)     __attribute__((invariant_assume(#COND)))
+
+/* Used to make identifiers for assert/assume annotations in a function. */
+#define STATIC_PASTE2(X,Y) X ## Y
+#define STATIC_PASTE1(X,Y) STATIC_PASTE2(X,Y)
+
+#define STATIC_ASSERT(COND)                          \
+  do {                                               \
+    __attribute__((assert_static(#COND), unused))    \
+    int STATIC_PASTE1(assert_static_, __COUNTER__);  \
+  } while(0)
+
+#define STATIC_ASSUME(COND)                          \
+  do {                                               \
+    __attribute__((assume_static(#COND), unused))    \
+    int STATIC_PASTE1(assume_static_, __COUNTER__);  \
+  } while(0)
+
+#define STATIC_ASSERT_RUNTIME(COND)                        \
+  do {                                                     \
+    __attribute__((assert_static_runtime(#COND), unused))  \
+    int STATIC_PASTE1(assert_static_runtime_, __COUNTER__); \
+  } while(0)
```

```
+
+#else /* XGILL_PLUGIN */
+
+#define STATIC_PRECONDITION(COND)          /* nothing */
+#define STATIC_PRECONDITION_ASSUME(COND)   /* nothing */
+#define STATIC_POSTCONDITION(COND)         /* nothing */
+#define STATIC_POSTCONDITION_ASSUME(COND)  /* nothing */
+#define STATIC_INVARIANT(COND)             /* nothing */
+#define STATIC_INVARIANT_ASSUME(COND)      /* nothing */
+
+#define STATIC_ASSERT(COND)          do { /* nothing */ } while(0)
+#define STATIC_ASSUME(COND)          do { /* nothing */ } while(0)
+#define STATIC_ASSERT_RUNTIME(COND)  do { /* nothing */ } while(0)
+
+#endif /* XGILL_PLUGIN */
+
+#define STATIC_SKIP_INFERENCE STATIC_INVARIANT(skip_inference())
+
+#endif /* HAVE_STATIC_ANNOTATIONS */
+
+#ifdef XGILL_PLUGIN
+
+/* Redefine runtime assertion macros to perform static assertions, for both
+ * debug and release builds. Don't include the original runtime assertions;
+ * this ensures the tool will consider cases where the assertion fails. */
+
+#undef TBB_NS_PRECONDITION
+#undef TBB_NS_ASSERTION
+#undef TBB_NS_POSTCONDITION
+
+#define TBB_NS_PRECONDITION(expr, str)   STATIC_ASSERT_RUNTIME(expr)
+#define TBB_NS_ASSERTION(expr, str)      STATIC_ASSERT_RUNTIME(expr)
+#define TBB_NS_POSTCONDITION(expr, str)  STATIC_ASSERT_RUNTIME(expr)
+
+#endif /* XGILL_PLUGIN */
+
+/*****************************************************************************
+** Macros for terminating execution when an unrecoverable condition is
+** reached.  These need to be compiled regardless of the DEBUG flag.
+*****************************************************************************/
+
+/**
+ * Terminate execution <i>immediately</i>, and if possible on the current
+ * platform, in such a way that execution can't be continued by other
+ * code (e.g., by intercepting a signal).
+ */
+#define TBB_NS_RUNTIMEABORT(msg)                                    \
+  NS_DebugBreak(NS_DEBUG_ABORT, msg, nullptr, __FILE__, __LINE__)
+
+
+/* Macros for checking the trueness of an expression passed in within an
```

```
+ * interface implementation.  These need to be compiled regardless of the */
+/* DEBUG flag
+*******************************************************************************/
+
+#define TBB_NS_ENSURE_TRUE(x, ret)                                    \
+  do {                                                          \
+    if (MOZ_UNLIKELY(!(x))) {                                   \
+       TBB_NS_WARNING("TBB_NS_ENSURE_TRUE(" #x ") failed");           \
+       return ret;                                              \
+    }                                                           \
+  } while(0)
+
+#define TBB_NS_ENSURE_FALSE(x, ret)                                   \
+  TBB_NS_ENSURE_TRUE(!(x), ret)
+
+#define TBB_NS_ENSURE_TRUE_VOID(x)                                    \
+  do {                                                          \
+    if (MOZ_UNLIKELY(!(x))) {                                   \
+       TBB_NS_WARNING("TBB_NS_ENSURE_TRUE(" #x ") failed");           \
+       return;                                                  \
+    }                                                           \
+  } while(0)
+
+#define TBB_NS_ENSURE_FALSE_VOID(x)                                   \
+  TBB_NS_ENSURE_TRUE_VOID(!(x))
+
+/*******************************************************************************
+** Macros for checking results
+*******************************************************************************/
+
+#if !defined(TOR_NASSERT) && !defined(XPCOM_GLUE_AVOID_NSPR)
+
+#define TBB_NS_ENSURE_SUCCESS_BODY(res, ret)                                \
+    char *msg = PR_smprintf("TBB_NS_ENSURE_SUCCESS(%s, %s) failed with "      \
+                            "result 0x%X", #res, #ret, __rv);             \
+    TBB_NS_WARNING(msg);                                                    \
+    PR_smprintf_free(msg);
+
+#define TBB_NS_ENSURE_SUCCESS_BODY_VOID(res)                                \
+    char *msg = PR_smprintf("TBB_NS_ENSURE_SUCCESS_VOID(%s) failed with "     \
+                            "result 0x%X", #res, __rv);                   \
+    TBB_NS_WARNING(msg);                                                    \
+    PR_smprintf_free(msg);
+
+#else
+
+#define TBB_NS_ENSURE_SUCCESS_BODY(res, ret)                                \
+    TBB_NS_WARNING("TBB_NS_ENSURE_SUCCESS(" #res ", " #ret ") failed");
+
+#define TBB_NS_ENSURE_SUCCESS_BODY_VOID(res)                                \
+    TBB_NS_WARNING("TBB_NS_ENSURE_SUCCESS_VOID(" #res ") failed");
```

```
+
+#endif
+
+#define TBB_NS_ENSURE_SUCCESS(res, ret)                                      \
+  do {                                                                       \
+    nsresult __rv = res; /* Don't evaluate |res| more than once */           \
+    if (TBB_NS_FAILED(__rv)) {                                               \
+      TBB_NS_ENSURE_SUCCESS_BODY(res, ret)                                   \
+      return ret;                                                            \
+    }                                                                        \
+  } while(0)
+
+#define TBB_NS_ENSURE_SUCCESS_VOID(res)                                      \
+  do {                                                                       \
+    nsresult __rv = res;                                                     \
+    if (TBB_NS_FAILED(__rv)) {                                               \
+      TBB_NS_ENSURE_SUCCESS_BODY_VOID(res)                                   \
+      return;                                                                \
+    }                                                                        \
+  } while(0)
+
+/****************************************************************************
+** Macros for checking state and arguments upon entering interface boundaries
+****************************************************************************/
+
+#define TBB_NS_ENSURE_ARG(arg)                                   \
+  TBB_NS_ENSURE_TRUE(arg, TBB_NS_ERROR_INVALID_ARG)
+
+#define TBB_NS_ENSURE_ARG_POINTER(arg)                           \
+  TBB_NS_ENSURE_TRUE(arg, TBB_NS_ERROR_INVALID_POINTER)
+
+#define TBB_NS_ENSURE_ARG_MIN(arg, min)                          \
+  TBB_NS_ENSURE_TRUE((arg) >= min, TBB_NS_ERROR_INVALID_ARG)
+
+#define TBB_NS_ENSURE_ARG_MAX(arg, max)                          \
+  TBB_NS_ENSURE_TRUE((arg) <= max, TBB_NS_ERROR_INVALID_ARG)
+
+#define TBB_NS_ENSURE_ARG_RANGE(arg, min, max)                   \
+  TBB_NS_ENSURE_TRUE(((arg) >= min) && ((arg) <= max), TBB_NS_ERROR_INVALID_ARG)
+
+#define TBB_NS_ENSURE_STATE(state)                               \
+  TBB_NS_ENSURE_TRUE(state, TBB_NS_ERROR_UNEXPECTED)
+
+#define TBB_NS_ENSURE_NO_AGGREGATION(outer)                      \
+  TBB_NS_ENSURE_FALSE(outer, TBB_NS_ERROR_NO_AGGREGATION)
+
+#define TBB_NS_ENSURE_PROPER_AGGREGATION(outer, iid)             \
+  TBB_NS_ENSURE_FALSE(outer && !iid.Equals(TBB_NS_GET_IID(nsISupports)),
    TBB_NS_ERROR_INVALID_ARG)
+
+/****************************************************************************/
```

```
+
+#ifdef XPCOM_GLUE
+  #define TBB_NS_CheckThreadSafe(owningThread, msg)
+#else
+  #define TBB_NS_CheckThreadSafe(owningThread, msg)                \
+    MOZ_ASSERT(owningThread == PR_GetCurrentThread(), msg)
+#endif
+
+/* When compiling the XPCOM Glue on Windows, we pretend that it's going to
+ * be linked with a static CRT (-MT) even when it's not. This means that we
+ * cannot link to data exports from the CRT, only function exports. So,
+ * instead of referencing "stderr" directly, use fdopen.
+ */
+#ifdef __cplusplus
+extern "C" {
+#endif
+
+NS_COM_GLUE void
+printf_stderr(const char *fmt, ...);
+
+#ifdef __cplusplus
+}
+#endif
+
+#endif /* nsDebugTor_h___ */
```

Listing 7: Sample Patch For Enabling Assertions In nsCOMPtr

E.3 JavaScript Engine Assertions

```
diff --git a/js/public/HashTable.h b/js/public/HashTable.h
index b9b7ef8..e44b5362 100644
--- a/js/public/HashTable.h
+++ b/js/public/HashTable.h
@@ -10,7 +10,7 @@
 #include "mozilla/Assertions.h"
 #include "mozilla/Attributes.h"
 #include "mozilla/Casting.h"
-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/PodOperations.h"
 #include "mozilla/TypeTraits.h"
 #include "mozilla/Util.h"
@@ -717,7 +717,7 @@ class HashTable : private AllocPolicy
     {
         friend class HashTable;
         HashNumber keyHash;
-        mozilla::DebugOnly<uint64_t> mutationCount;
+        mozilla::DebugOnlyTor<uint64_t> mutationCount;

         AddPtr(Entry &entry, HashNumber hn) : Ptr(entry), keyHash(hn) {}
       public:
@@ -740,7 +740,7 @@ class HashTable : private AllocPolicy
         }

         Entry *cur, *end;
-        mozilla::DebugOnly<bool> validEntry;
+        mozilla::DebugOnlyTor<bool> validEntry;

       public:
         Range() : cur(NULL), end(NULL), validEntry(false) {}
@@ -877,8 +877,8 @@ class HashTable : private AllocPolicy
 #endif

     friend class js::ReentrancyGuard;
-    mutable mozilla::DebugOnly<bool> entered;
-    mozilla::DebugOnly<uint64_t>     mutationCount;
+    mutable mozilla::DebugOnlyTor<bool> entered;
+    mozilla::DebugOnlyTor<uint64_t>     mutationCount;

     // The default initial capacity is 32 (enough to hold 16 elements), but it
     // can be as low as 4.
diff --git a/js/public/Utility.h b/js/public/Utility.h
index 7582673..ba997fb 100644
--- a/js/public/Utility.h
+++ b/js/public/Utility.h
@@ -7,7 +7,7 @@
 #ifndef js_Utility_h
```

```
 #define js_Utility_h

-#include "mozilla/Assertions.h"
+#include "mozilla/AssertionsTor.h"
 #include "mozilla/Attributes.h"
 #include "mozilla/Compiler.h"
 #include "mozilla/Scoped.h"
@@ -39,11 +39,11 @@ namespace js {}
  */
 #define JS_FREE_PATTERN 0xDA

-#define JS_ASSERT(expr)           MOZ_ASSERT(expr)
-#define JS_ASSERT_IF(cond, expr)  MOZ_ASSERT_IF(cond, expr)
-#define JS_NOT_REACHED(reason)    MOZ_NOT_REACHED(reason)
-#define JS_ALWAYS_TRUE(expr)      MOZ_ALWAYS_TRUE(expr)
-#define JS_ALWAYS_FALSE(expr)     MOZ_ALWAYS_FALSE(expr)
+#define JS_ASSERT(expr)           TBB_MOZ_ASSERT(expr)
+#define JS_ASSERT_IF(cond, expr)  TBB_MOZ_ASSERT_IF(cond, expr)
+#define JS_NOT_REACHED(reason)    TBB_MOZ_NOT_REACHED(reason)
+#define JS_ALWAYS_TRUE(expr)      TBB_MOZ_ALWAYS_TRUE(expr)
+#define JS_ALWAYS_FALSE(expr)     TBB_MOZ_ALWAYS_FALSE(expr)

 #ifdef DEBUG
 # ifdef JS_THREADSAFE
@@ -56,15 +56,15 @@ namespace js {}
 #endif

 #if defined(DEBUG)
-# define JS_DIAGNOSTICS_ASSERT(expr) MOZ_ASSERT(expr)
+# define JS_DIAGNOSTICS_ASSERT(expr) TBB_MOZ_ASSERT(expr)
 #elif defined(JS_CRASH_DIAGNOSTICS)
-# define JS_DIAGNOSTICS_ASSERT(expr) do { if (!(expr)) MOZ_CRASH(); } while(0)
+# define JS_DIAGNOSTICS_ASSERT(expr) do { if (!(expr)) TBB_MOZ_CRASH(); } while(0)
 #else
 # define JS_DIAGNOSTICS_ASSERT(expr) ((void) 0)
 #endif

-#define JS_STATIC_ASSERT(cond)           MOZ_STATIC_ASSERT(cond, "JS_STATIC_ASSERT")
-#define JS_STATIC_ASSERT_IF(cond, expr)  MOZ_STATIC_ASSERT_IF(cond, expr, "
    JS_STATIC_ASSERT_IF")
+#define JS_STATIC_ASSERT(cond)           TBB_MOZ_STATIC_ASSERT(cond, "
    JS_STATIC_ASSERT")
+#define JS_STATIC_ASSERT_IF(cond, expr)  TBB_MOZ_STATIC_ASSERT_IF(cond, expr, "
    JS_STATIC_ASSERT_IF")

 extern MOZ_NORETURN JS_PUBLIC_API(void)
 JS_Assert(const char *s, const char *file, int ln);
diff --git a/js/public/Vector.h b/js/public/Vector.h
index 8982ad3..71a3372 100644
--- a/js/public/Vector.h
+++ b/js/public/Vector.h
```

```
@@ -251,13 +251,13 @@ class Vector : private AllocPolicy
     T *mBegin;
     size_t mLength;     /* Number of elements in the Vector. */
     size_t mCapacity;   /* Max number of elements storable in the Vector without
         resizing. */
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     size_t mReserved;   /* Max elements of reserved or used space in this vector. */
 #endif

     mozilla::AlignedStorage<sInlineBytes> storage;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     friend class ReentrancyGuard;
     bool entered;
 #endif
@@ -287,7 +287,7 @@ class Vector : private AllocPolicy
         return mBegin + mLength;
     }

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     size_t reserved() const {
         JS_ASSERT(mReserved <= mCapacity);
         JS_ASSERT(mLength <= mReserved);
@@ -530,7 +530,7 @@ JS_ALWAYS_INLINE
 Vector<T,N,AllocPolicy>::Vector(AllocPolicy ap)
   : AllocPolicy(ap), mBegin((T *)storage.addr()), mLength(0),
     mCapacity(sInlineCapacity)
-#ifdef DEBUG
+#ifndef TOR_NASSERT
   , mReserved(sInlineCapacity), entered(false)
 #endif
 {}
@@ -540,13 +540,13 @@ template <class T, size_t N, class AllocPolicy>
 JS_ALWAYS_INLINE
 Vector<T, N, AllocPolicy>::Vector(MoveRef<Vector> rhs)
     : AllocPolicy(rhs)
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     , entered(false)
 #endif
 {
     mLength = rhs->mLength;
     mCapacity = rhs->mCapacity;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     mReserved = rhs->mReserved;
 #endif

```

```
@@ -567,7 +567,7 @@ Vector<T, N, AllocPolicy>::Vector(MoveRef<Vector> rhs)
          rhs->mBegin = (T *) rhs->storage.addr();
          rhs->mCapacity = sInlineCapacity;
          rhs->mLength = 0;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
          rhs->mReserved = sInlineCapacity;
 #endif
      }
@@ -714,7 +714,7 @@ Vector<T,N,AP>::initCapacity(size_t request)
          return false;
      mBegin = newbuf;
      mCapacity = request;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
      mReserved = request;
 #endif
      return true;
@@ -728,7 +728,7 @@ Vector<T,N,AP>::reserve(size_t request)
          return false;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
      if (request > mReserved)
          mReserved = request;
      JS_ASSERT(mLength <= mReserved);
@@ -761,7 +761,7 @@ Vector<T,N,AP>::growByImpl(size_t incr)
          Impl::initialize(endNoCheck(), newend);
      mLength += incr;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
      if (mLength > mReserved)
          mReserved = mLength;
 #endif
@@ -826,7 +826,7 @@ Vector<T,N,AP>::clearAndFree()
      mBegin = (T *)storage.addr();
      mCapacity = sInlineCapacity;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
      mReserved = sInlineCapacity;
 #endif
 }
@@ -847,7 +847,7 @@ Vector<T,N,AP>::append(U t)
          return false;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
```

```
198      if (mLength + 1 > mReserved)
199          mReserved = mLength + 1;
200  #endif
201 @@ -874,7 +874,7 @@ Vector<T,N,AP>::appendN(const T &t, size_t needed)
202      if (mLength + needed > mCapacity && !growStorageBy(needed))
203          return false;
204
205 -#ifdef DEBUG
206 +#ifndef TOR_NASSERT
207      if (mLength + needed > mReserved)
208          mReserved = mLength + needed;
209  #endif
210 @@ -936,7 +936,7 @@ Vector<T,N,AP>::append(const U *insBegin, const U *insEnd)
211      if (mLength + needed > mCapacity && !growStorageBy(needed))
212          return false;
213
214 -#ifdef DEBUG
215 +#ifndef TOR_NASSERT
216      if (mLength + needed > mReserved)
217          mReserved = mLength + needed;
218  #endif
219 @@ -1016,7 +1016,7 @@ Vector<T,N,AP>::extractRawBuffer()
220          mBegin = (T *)storage.addr();
221          mLength = 0;
222          mCapacity = sInlineCapacity;
223 -#ifdef DEBUG
224 +#ifndef TOR_NASSERT
225          mReserved = sInlineCapacity;
226  #endif
227      }
228 @@ -1052,7 +1052,7 @@ Vector<T,N,AP>::replaceRawBuffer(T *p, size_t aLength)
229          mLength = aLength;
230          mCapacity = aLength;
231      }
232 -#ifdef DEBUG
233 +#ifndef TOR_NASSERT
234      mReserved = aLength;
235  #endif
236  }
237 @@ -1093,7 +1093,7 @@ Vector<T,N,AP>::swap(Vector &other)
238
239      Swap(mLength, other.mLength);
240      Swap(mCapacity, other.mCapacity);
241 -#ifdef DEBUG
242 +#ifndef TOR_NASSERT
243      Swap(mReserved, other.mReserved);
244  #endif
245  }
246 diff --git a/js/src/assembler/assembler/LinkBuffer.h b/js/src/assembler/assembler/
        LinkBuffer.h
247 index 8891232..f176dcb 100644
```

```
--- a/js/src/assembler/assembler/LinkBuffer.h
+++ b/js/src/assembler/assembler/LinkBuffer.h
@@ -70,7 +70,7 @@ public:
         m_code = executableAllocAndCopy(*masm, executableAllocator, poolp);
         m_executablePool = *poolp;
         m_size = masm->m_assembler.size();  // must come after call to
             executableAllocAndCopy()!
-#ifndef NDEBUG
+#ifndef TOR_NASSERT
         m_completed = false;
 #endif
         *ok = !!m_code;
@@ -81,7 +81,7 @@ public:
         , m_code(NULL)
         , m_size(0)
         , m_codeKind(kind)
-#ifndef NDEBUG
+#ifndef TOR_NASSERT
         , m_completed(false)
 #endif
     {
@@ -92,7 +92,7 @@ public:
         , m_code(ncode)
         , m_size(size)
         , m_codeKind(kind)
-#ifndef NDEBUG
+#ifndef TOR_NASSERT
         , m_completed(false)
 #endif
     {
@@ -208,7 +208,7 @protected:

     void performFinalization()
     {
-#ifndef NDEBUG
+#ifndef TOR_NASSERT
         ASSERT(!m_completed);
         m_completed = true;
 #endif
@@ -221,7 +221,7 @@ protected:
     void* m_code;
     size_t m_size;
     CodeKind m_codeKind;
-#ifndef NDEBUG
+#ifndef TOR_NASSERT
     bool m_completed;
 #endif
 };
diff --git a/js/src/assembler/assembler/MacroAssemblerX86Common.h b/js/src/assembler/
    assembler/MacroAssemblerX86Common.h
index 8781642..7f7a291 100644
```

```
--- a/js/src/assembler/assembler/MacroAssemblerX86Common.h
+++ b/js/src/assembler/assembler/MacroAssemblerX86Common.h
@@ -1449,7 +1449,7 @@ private:


 #endif // PLATFORM(MAC)
-#elif !defined(NDEBUG) // CPU(X86)
+#elif !defined(TOR_NASSERT) // CPU(X86)

     // On x86-64 we should never be checking for SSE2 in a non-debug build,
     // but non debug add this method to keep the asserts above happy.
diff --git a/js/src/assembler/assembler/MacroAssemblerX86_64.h b/js/src/assembler/
    assembler/MacroAssemblerX86_64.h
index c76b6ad..459b49a 100644
--- a/js/src/assembler/assembler/MacroAssemblerX86_64.h
+++ b/js/src/assembler/assembler/MacroAssemblerX86_64.h
@@ -30,7 +30,7 @@
 #ifndef assembler_assembler_MacroAssemblerX86_64_h
 #define assembler_assembler_MacroAssemblerX86_64_h

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "assembler/wtf/Platform.h"

@@ -126,7 +126,7 @@ public:

     Call call()
     {
-        mozilla::DebugOnly<DataLabelPtr> label = moveWithPatch(ImmPtr(0),
    scratchRegister);
+        mozilla::DebugOnlyTor<DataLabelPtr> label = moveWithPatch(ImmPtr(0),
    scratchRegister);
         Call result = Call(m_assembler.call(scratchRegister), Call::Linkable);
         ASSERT(differenceBetween(label, result) == REPTACH_OFFSET_CALL_R11);
         return result;
@@ -134,7 +134,7 @@ public:

     Call tailRecursiveCall()
     {
-        mozilla::DebugOnly<DataLabelPtr> label = moveWithPatch(ImmPtr(0),
    scratchRegister);
+        mozilla::DebugOnlyTor<DataLabelPtr> label = moveWithPatch(ImmPtr(0),
    scratchRegister);
         Jump newJump = Jump(m_assembler.jmp_r(scratchRegister));
         ASSERT(differenceBetween(label, newJump) == REPTACH_OFFSET_CALL_R11);
         return Call::fromTailJump(newJump);
@@ -143,7 +143,7 @@ public:
     Call makeTailRecursiveCall(Jump oldJump)
     {
         oldJump.link(this);
```

```
-        mozilla::DebugOnly<DataLabelPtr> label = moveWithPatch(ImmPtr(0),
    scratchRegister);
+        mozilla::DebugOnlyTor<DataLabelPtr> label = moveWithPatch(ImmPtr(0),
    scratchRegister);
         Jump newJump = Jump(m_assembler.jmp_r(scratchRegister));
         ASSERT(differenceBetween(label, newJump) == REPTACH_OFFSET_CALL_R11);
         return Call::fromTailJump(newJump);
diff --git a/js/src/assembler/wtf/Assertions.h b/js/src/assembler/wtf/Assertions.h
index eb0744e..df4948b 100644
--- a/js/src/assembler/wtf/Assertions.h
+++ b/js/src/assembler/wtf/Assertions.h
@@ -27,9 +27,9 @@
 #define assembler_wtf_Assertions_h

 #include "Platform.h"
-#include "mozilla/Assertions.h"
+#include "mozilla/AssertionsTor.h"

-#ifndef DEBUG
+#ifdef TOR_NASSERT
     /*
      * Prevent unused-variable warnings by defining the macro WTF uses to test
      * for assertions taking effect.
@@ -37,13 +37,13 @@
 #  define ASSERT_DISABLED 1
 #endif

-#define ASSERT(assertion) MOZ_ASSERT(assertion)
+#define ASSERT(assertion) TBB_MOZ_ASSERT(assertion)
 #define ASSERT_UNUSED(variable, assertion) do { \
     (void)variable; \
     ASSERT(assertion); \
 } while (0)
-#define ASSERT_NOT_REACHED() MOZ_NOT_REACHED("")
-#define CRASH() MOZ_CRASH()
-#define COMPILE_ASSERT(exp, name) MOZ_STATIC_ASSERT(exp, #name)
+#define ASSERT_NOT_REACHED() TBB_MOZ_NOT_REACHED("")
+#define CRASH() TBB_MOZ_CRASH()
+#define COMPILE_ASSERT(exp, name) TBB_MOZ_STATIC_ASSERT(exp, #name)

 #endif /* assembler_wtf_Assertions_h */
diff --git a/js/src/ctypes/CTypes.h b/js/src/ctypes/CTypes.h
index 39a00ee..89fce64 100644
--- a/js/src/ctypes/CTypes.h
+++ b/js/src/ctypes/CTypes.h
@@ -6,7 +6,7 @@
 #ifndef ctypes_CTypes_h
 #define ctypes_CTypes_h

-#include "mozilla/Assertions.h"
+#include "mozilla/AssertionsTor.h"
```

```
 #include "mozilla/TypeTraits.h"

 #include "jscntxt.h"
@@ -60,7 +60,7 @@ private:
 template<class T, size_t N = 0>
 class Array : public Vector<T, N, SystemAllocPolicy>
 {
-  MOZ_STATIC_ASSERT((!mozilla::IsSame<T, JS::Value>::value),
+  TBB_MOZ_STATIC_ASSERT((!mozilla::IsSame<T, JS::Value>::value),
                     "use JS::AutoValueVector instead");
 };

diff --git a/js/src/ds/LifoAlloc.h b/js/src/ds/LifoAlloc.h
index 3e663e4..8258d9d 100644
--- a/js/src/ds/LifoAlloc.h
+++ b/js/src/ds/LifoAlloc.h
@@ -7,7 +7,7 @@
 #ifndef ds_LifoAlloc_h
 #define ds_LifoAlloc_h

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/MemoryChecking.h"
 #include "mozilla/PodOperations.h"
 #include "mozilla/TypeTraits.h"
@@ -261,7 +261,7 @@ class LifoAlloc
         if (latest && (result = latest->tryAlloc(n)))
             return result;

-        mozilla::DebugOnly<BumpChunk *> chunk = getOrCreateChunk(n);
+        mozilla::DebugOnlyTor<BumpChunk *> chunk = getOrCreateChunk(n);
         JS_ASSERT(chunk);

         return latest->allocInfallible(n);
diff --git a/js/src/frontend/BytecodeEmitter.cpp b/js/src/frontend/BytecodeEmitter.
    cpp
index bf8d240..1f3b10c 100644
--- a/js/src/frontend/BytecodeEmitter.cpp
+++ b/js/src/frontend/BytecodeEmitter.cpp
@@ -10,7 +10,7 @@

 #include "frontend/BytecodeEmitter-inl.h"

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/FloatingPoint.h"
 #include "mozilla/PodOperations.h"

@@ -43,7 +43,7 @@ using namespace js;
 using namespace js::gc;
 using namespace js::frontend;
```

```
442
443 -using mozilla::DebugOnly;
444 +using mozilla::DebugOnlyTor;
445  using mozilla::DoubleIsInt32;
446  using mozilla::PodCopy;
447
448 @@ -1389,7 +1389,7 @@ BindNameToSlotHelper(JSContext *cx, BytecodeEmitter *bce,
        ParseNode *pn)
449          if (dn->pn_cookie.level() != bce->script->staticLevel)
450              return true;
451
452 -        DebugOnly<JSFunction *> fun = bce->sc->asFunctionBox()->function();
453 +        DebugOnlyTor<JSFunction *> fun = bce->sc->asFunctionBox()->function();
454          JS_ASSERT(fun->isLambda());
455          JS_ASSERT(pn->pn_atom == fun->atom());
456
457 @@ -2841,7 +2841,7 @@ EmitDestructuringOpsHelper(JSContext *cx, BytecodeEmitter *bce,
       ParseNode *pn,
458      ParseNode *pn2, *pn3;
459      bool doElemOp;
460
461 -#ifdef DEBUG
462 +#ifndef TOR_NASSERT
463      int stackDepth = bce->stackDepth;
464      JS_ASSERT(stackDepth != 0);
465      JS_ASSERT(pn->isArity(PN_LIST));
466 @@ -4065,7 +4065,7 @@ EmitLet(JSContext *cx, BytecodeEmitter *bce, ParseNode *pnLet)
467      StmtInfoBCE stmtInfo(cx);
468      PushBlockScopeBCE(bce, &stmtInfo, *blockObj, bce->offset());
469
470 -    DebugOnly<ptrdiff_t> bodyBegin = bce->offset();
471 +    DebugOnlyTor<ptrdiff_t> bodyBegin = bce->offset();
472      if (!EmitEnterBlock(cx, bce, letBody, JSOP_ENTERLET0))
473          return false;
474
475 @@ -4076,7 +4076,7 @@ EmitLet(JSContext *cx, BytecodeEmitter *bce, ParseNode *pnLet)
476      JS_ASSERT(leaveOp == JSOP_LEAVEBLOCK || leaveOp == JSOP_LEAVEBLOCKEXPR);
477      EMIT_UINT16_IMM_OP(leaveOp, blockObj->slotCount());
478
479 -    DebugOnly<ptrdiff_t> bodyEnd = bce->offset();
480 +    DebugOnlyTor<ptrdiff_t> bodyEnd = bce->offset();
481      JS_ASSERT(bodyEnd > bodyBegin);
482
483      return PopStatementBCE(cx, bce);
484 @@ -4223,7 +4223,7 @@ EmitForIn(JSContext *cx, BytecodeEmitter *bce, ParseNode *pn,
       ptrdiff_t top)
485      if (EmitLoopHead(cx, bce, NULL) < 0)
486          return false;
487
488 -#ifdef DEBUG
489 +#ifndef TOR_NASSERT
```

```
     int loopDepth = bce->stackDepth;
 #endif

diff --git a/js/src/frontend/TokenStream.cpp b/js/src/frontend/TokenStream.cpp
index 02da46f..b2aada3 100644
--- a/js/src/frontend/TokenStream.cpp
+++ b/js/src/frontend/TokenStream.cpp
@@ -918,7 +918,7 @@ TokenStream::atomize(JSContext *cx, CharBuffer &cb)
     return AtomizeChars<CanGC>(cx, cb.begin(), cb.length());
 }

-#ifdef DEBUG
+#ifndef TOR_NASSERT
 bool
 IsTokenSane(Token *tp)
 {
diff --git a/js/src/frontend/TokenStream.h b/js/src/frontend/TokenStream.h
index 48fdec3..f279eff2 100644
--- a/js/src/frontend/TokenStream.h
+++ b/js/src/frontend/TokenStream.h
@@ -11,7 +11,7 @@
  * JS lexical scanner interface.
  */

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/PodOperations.h"

 #include <stddef.h>
@@ -883,7 +883,7 @@ class MOZ_STACK_CLASS TokenStream
     }

     void consumeKnownChar(int32_t expect) {
-        mozilla::DebugOnly<int32_t> c = getChar();
+        mozilla::DebugOnlyTor<int32_t> c = getChar();
         JS_ASSERT(c == expect);
     }

diff --git a/js/src/gc/Heap.h b/js/src/gc/Heap.h
index 4f04ace..7d571c3 100644
--- a/js/src/gc/Heap.h
+++ b/js/src/gc/Heap.h
@@ -100,7 +100,7 @@ struct Cell
     inline JSRuntime *runtime() const;
     inline Zone *tenuredZone() const;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     inline bool isAligned() const;
     inline bool isTenured() const;
 #endif
```

```
@@ -994,7 +994,7 @@ Cell::tenuredZone() const
     return arenaHeader()->zone;
 }

-#ifdef DEBUG
+#ifndef TOR_NASSERT
 bool
 Cell::isAligned() const
 {
diff --git a/js/src/gc/Marking.cpp b/js/src/gc/Marking.cpp
index 47a7fca..df55b17 100644
--- a/js/src/gc/Marking.cpp
+++ b/js/src/gc/Marking.cpp
@@ -6,7 +6,7 @@

 #include "gc/Marking.h"

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "jit/IonCode.h"
 #include "vm/Shape.h"
@@ -20,7 +20,7 @@
 using namespace js;
 using namespace js::gc;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;

 void * const js::NullPtr::constNullValue = NULL;

@@ -126,7 +126,7 @@ CheckMarkedThing(JSTracer *trc, T *thing)
     JS_ASSERT(thing->zone()->rt == trc->runtime);
     JS_ASSERT(trc->debugPrinter || trc->debugPrintArg);

-    DebugOnly<JSRuntime *> rt = trc->runtime;
+    DebugOnlyTor<JSRuntime *> rt = trc->runtime;

     JS_ASSERT_IF(IS_GC_MARKING_TRACER(trc) && rt->gcManipulatingDeadZones,
                  !thing->zone()->scheduledForDestruction);
@@ -378,7 +378,7 @@ gc::MarkKind(JSTracer *trc, void **thingp, JSGCTraceKind kind)
 {
     JS_ASSERT(thingp);
     JS_ASSERT(*thingp);
-    DebugOnly<Cell *> cell = static_cast<Cell *>(*thingp);
+    DebugOnlyTor<Cell *> cell = static_cast<Cell *>(*thingp);
     JS_ASSERT_IF(cell->isTenured(), kind == MapAllocToTraceKind(cell->
         tenuredGetAllocKind()));
     switch (kind) {
       case JSTRACE_OBJECT:
diff --git a/js/src/gc/RootMarking.cpp b/js/src/gc/RootMarking.cpp
```

```
index 861c2d6..ad116b4 100644
--- a/js/src/gc/RootMarking.cpp
+++ b/js/src/gc/RootMarking.cpp
@@ -4,7 +4,7 @@
  * License, v. 2.0. If a copy of the MPL was not distributed with this
  * file, You can obtain one at http://mozilla.org/MPL/2.0/. */

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/Util.h"

 #include "jsapi.h"
@@ -476,7 +476,7 @@ AutoGCRooter::trace(JSTracer *trc)
       case OBJOBJHASHMAP: {
         AutoObjectObjectHashMap::HashMapImpl &map = static_cast<
             AutoObjectObjectHashMap *>(this)->map;
         for (AutoObjectObjectHashMap::Enum e(map); !e.empty(); e.popFront()) {
-            mozilla::DebugOnly<JSObject *> key = e.front().key;
+            mozilla::DebugOnlyTor<JSObject *> key = e.front().key;
             MarkObjectRoot(trc, const_cast<JSObject **>(&e.front().key), "
                 AutoObjectObjectHashMap key");
             JS_ASSERT(key == e.front().key);  // Needs rewriting for moving GC, see
                 bug 726687.
             MarkObjectRoot(trc, &e.front().value, "AutoObjectObjectHashMap value");
@@ -488,7 +488,7 @@ AutoGCRooter::trace(JSTracer *trc)
         AutoObjectUnsigned32HashMap *self = static_cast<AutoObjectUnsigned32HashMap
             *>(this);
         AutoObjectUnsigned32HashMap::HashMapImpl &map = self->map;
         for (AutoObjectUnsigned32HashMap::Enum e(map); !e.empty(); e.popFront()) {
-            mozilla::DebugOnly<JSObject *> key = e.front().key;
+            mozilla::DebugOnlyTor<JSObject *> key = e.front().key;
             MarkObjectRoot(trc, const_cast<JSObject **>(&e.front().key), "
                 AutoObjectUnsignedHashMap key");
             JS_ASSERT(key == e.front().key);  // Needs rewriting for moving GC, see
                 bug 726687.
         }
@@ -499,7 +499,7 @@ AutoGCRooter::trace(JSTracer *trc)
         AutoObjectHashSet *self = static_cast<AutoObjectHashSet *>(this);
         AutoObjectHashSet::HashSetImpl &set = self->set;
         for (AutoObjectHashSet::Enum e(set); !e.empty(); e.popFront()) {
-            mozilla::DebugOnly<JSObject *> obj = e.front();
+            mozilla::DebugOnlyTor<JSObject *> obj = e.front();
             MarkObjectRoot(trc, const_cast<JSObject **>(&e.front()), "
                 AutoObjectHashSet value");
             JS_ASSERT(obj == e.front());  // Needs rewriting for moving GC, see bug
                 726687.
         }
diff --git a/js/src/jit/AsmJS.cpp b/js/src/jit/AsmJS.cpp
index d05289e..a42c81f 100644
--- a/js/src/jit/AsmJS.cpp
+++ b/js/src/jit/AsmJS.cpp
```

```
@@ -1089,7 +1089,7 @@ class MOZ_STACK_CLASS ModuleCompiler

     TokenStream &                   tokenStream_;

-    DebugOnly<int>                  currentPass_;
+    DebugOnlyTor<int>                  currentPass_;

     bool addStandardLibraryMathName(const char *name, AsmJSMathBuiltin builtin) {
         JSAtom *atom = Atomize(cx_, name, strlen(name));
diff --git a/js/src/jit/BacktrackingAllocator.cpp b/js/src/jit/BacktrackingAllocator.cpp
index 55dbdfb..61b2324 100644
--- a/js/src/jit/BacktrackingAllocator.cpp
+++ b/js/src/jit/BacktrackingAllocator.cpp
@@ -9,7 +9,7 @@
 using namespace js;
 using namespace js::jit;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;

 bool
 BacktrackingAllocator::init()
@@ -1117,7 +1117,7 @@ BacktrackingAllocator::populateSafepoints()
             // is not used with gcthings or nunboxes, or we would have to add the input reg
             // to this safepoint.
             if (ins == reg->ins() && !reg->isTemp()) {
-                DebugOnly<LDefinition*> def = reg->def();
+                DebugOnlyTor<LDefinition*> def = reg->def();
                 JS_ASSERT_IF(def->policy() == LDefinition::MUST_REUSE_INPUT,
                              def->type() == LDefinition::GENERAL || def->type() == LDefinition::DOUBLE);
                 continue;
diff --git a/js/src/jit/BaselineIC.cpp b/js/src/jit/BaselineIC.cpp
index 9652169..150dc3c 100644
--- a/js/src/jit/BaselineIC.cpp
+++ b/js/src/jit/BaselineIC.cpp
@@ -601,7 +601,7 @@ void
 ICStubCompiler::enterStubFrame(MacroAssembler &masm, Register scratch)
 {
     EmitEnterStubFrame(masm, scratch);
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     entersStubFrame_ = true;
 #endif
 }
@@ -992,7 +992,7 @@ DoProfilerFallback(JSContext *cx, BaselineFrame *frame, ICProfiler_Fallback *stu
 {
     RootedScript script(cx, frame->script());
```

```
681      RootedFunction func(cx, frame->maybeFun());
682 -    mozilla::DebugOnly<ICEntry *> icEntry = stub->icEntry();
683 +    mozilla::DebugOnlyTor<ICEntry *> icEntry = stub->icEntry();
684
685      FallbackICSpew(cx, stub, "Profiler");
686
687 @@ -4910,7 +4910,7 @@ DoGetNameFallback(JSContext *cx, BaselineFrame *frame,
     ICGetName_Fallback *stub,
688  {
689      RootedScript script(cx, frame->script());
690      jsbytecode *pc = stub->icEntry()->pc(script);
691 -    mozilla::DebugOnly<JSOp> op = JSOp(*pc);
692 +    mozilla::DebugOnlyTor<JSOp> op = JSOp(*pc);
693      FallbackICSpew(cx, stub, "GetName(%s)", js_CodeName[JSOp(*pc)]);
694
695      JS_ASSERT(op == JSOP_NAME || op == JSOP_CALLNAME || op == JSOP_GETGNAME || op ==
           JSOP_CALLGNAME);
696 @@ -5043,7 +5043,7 @@ DoBindNameFallback(JSContext *cx, BaselineFrame *frame,
     ICBindName_Fallback *stu
697                      HandleObject scopeChain, MutableHandleValue res)
698  {
699      jsbytecode *pc = stub->icEntry()->pc(frame->script());
700 -    mozilla::DebugOnly<JSOp> op = JSOp(*pc);
701 +    mozilla::DebugOnlyTor<JSOp> op = JSOp(*pc);
702      FallbackICSpew(cx, stub, "BindName(%s)", js_CodeName[JSOp(*pc)]);
703
704      JS_ASSERT(op == JSOP_BINDNAME);
705 @@ -5087,7 +5087,7 @@ DoGetIntrinsicFallback(JSContext *cx, BaselineFrame *frame,
     ICGetIntrinsic_Fallb
706  {
707      RootedScript script(cx, frame->script());
708      jsbytecode *pc = stub->icEntry()->pc(script);
709 -    mozilla::DebugOnly<JSOp> op = JSOp(*pc);
710 +    mozilla::DebugOnlyTor<JSOp> op = JSOp(*pc);
711      FallbackICSpew(cx, stub, "GetIntrinsic(%s)", js_CodeName[JSOp(*pc)]);
712
713      JS_ASSERT(op == JSOP_GETINTRINSIC || op == JSOP_CALLINTRINSIC);
714 diff --git a/js/src/jit/BaselineIC.h b/js/src/jit/BaselineIC.h
715 index 63da318..2d13e75 100644
716 --- a/js/src/jit/BaselineIC.h
717 +++ b/js/src/jit/BaselineIC.h
718 @@ -980,7 +980,7 @@ class ICStubCompiler
719      // Prevent GC in the middle of stub compilation.
720      js::gc::AutoSuppressGC suppressGC;
721
722 -    mozilla::DebugOnly<bool> entersStubFrame_;
723 +    mozilla::DebugOnlyTor<bool> entersStubFrame_;
724
725    protected:
726      JSContext *cx;
727 diff --git a/js/src/jit/BaselineInspector.h b/js/src/jit/BaselineInspector.h
```

```
index bb40c3a..72035b1 100644
--- a/js/src/jit/BaselineInspector.h
+++ b/js/src/jit/BaselineInspector.h
@@ -67,7 +67,7 @@ class BaselineInspector
     }

   private:
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     bool isValidPC(jsbytecode *pc) {
         return (pc >= script->code) && (pc < script->code + script->length);
     }
diff --git a/js/src/jit/BaselineJIT.cpp b/js/src/jit/BaselineJIT.cpp
index b3832f0..f6b0bd1 100644
--- a/js/src/jit/BaselineJIT.cpp
+++ b/js/src/jit/BaselineJIT.cpp
@@ -35,7 +35,7 @@ BaselineScript::BaselineScript(uint32_t prologueOffset, uint32_t
    spsPushToggleOf
   : method_(NULL),
     fallbackStubSpace_(),
     prologueOffset_(prologueOffset),
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     spsOn_(false),
 #endif
     spsPushToggleOffset_(spsPushToggleOffset),
@@ -757,7 +757,7 @@ BaselineScript::toggleSPS(bool enable)
         Assembler::ToggleToCmp(pushToggleLocation);
     else
         Assembler::ToggleToJmp(pushToggleLocation);
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     spsOn_ = enable;
 #endif
 }
diff --git a/js/src/jit/BaselineJIT.h b/js/src/jit/BaselineJIT.h
index c3f9981..5db487f 100644
--- a/js/src/jit/BaselineJIT.h
+++ b/js/src/jit/BaselineJIT.h
@@ -110,8 +110,8 @@ struct BaselineScript
     uint32_t prologueOffset_;

     // The offsets for the toggledJump instructions for SPS update ICs.
-#ifdef DEBUG
-    mozilla::DebugOnly<bool> spsOn_;
+#ifndef TOR_NASSERT
+    mozilla::DebugOnlyTor<bool> spsOn_;
 #endif
     uint32_t spsPushToggleOffset_;

diff --git a/js/src/jit/CodeGenerator.cpp b/js/src/jit/CodeGenerator.cpp
```

```
index 534ae07..5d263d2 100644
--- a/js/src/jit/CodeGenerator.cpp
+++ b/js/src/jit/CodeGenerator.cpp
@@ -4,9 +4,9 @@
  * License, v. 2.0. If a copy of the MPL was not distributed with this
  * file, You can obtain one at http://mozilla.org/MPL/2.0/. */

-#include "mozilla/Assertions.h"
+#include "mozilla/AssertionsTor.h"
 #include "mozilla/Attributes.h"
-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/Util.h"

 #include "PerfSpewer.h"
@@ -32,7 +32,7 @@
 using namespace js;
 using namespace js::jit;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;
 using mozilla::Maybe;

 namespace js {
@@ -317,19 +317,19 @@ class OutOfLineTestObject : public OutOfLineCodeBase<
    CodeGenerator>
     Label *ifTruthy_;
     Label *ifFalsy_;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     bool initialized() { return ifTruthy_ != NULL; }
 #endif

   public:
     OutOfLineTestObject()
-#ifdef DEBUG
+#ifndef TOR_NASSERT
       : ifTruthy_(NULL), ifFalsy_(NULL)
 #endif
     { }

     bool accept(CodeGenerator *codegen) MOZ_FINAL MOZ_OVERRIDE {
-        MOZ_ASSERT(initialized());
+        TBB_MOZ_ASSERT(initialized());
         codegen->emitOOLTestObject(objreg_, ifTruthy_, ifFalsy_, scratch_);
         return true;
     }
@@ -338,8 +338,8 @@ class OutOfLineTestObject : public OutOfLineCodeBase<
    CodeGenerator>
     // jump to if the object is truthy or falsy, and a scratch register for
```

```
827      // use in the out-of-line path.
828      void setInputAndTargets(Register objreg, Label *ifTruthy, Label *ifFalsy,
             Register scratch) {
829  -       MOZ_ASSERT(!initialized());
830  -       MOZ_ASSERT(ifTruthy);
831  +       TBB_MOZ_ASSERT(!initialized());
832  +       TBB_MOZ_ASSERT(ifTruthy);
833          objreg_ = objreg;
834          scratch_ = scratch;
835          ifTruthy_ = ifTruthy;
836  @@ -438,7 +438,7 @@ CodeGenerator::testValueTruthy(const ValueOperand &value,
837   bool
838   CodeGenerator::visitTestOAndBranch(LTestOAndBranch *lir)
839   {
840  -    MOZ_ASSERT(lir->mir()->operandMightEmulateUndefined(),
841  +    TBB_MOZ_ASSERT(lir->mir()->operandMightEmulateUndefined(),
842                  "Objects which can't emulate undefined should have been constant-
                       folded");
843
844      OutOfLineTestObject *ool = new OutOfLineTestObject();
845  @@ -516,7 +516,7 @@ CodeGenerator::visitTypeObjectDispatch(LTypeObjectDispatch *lir)
846          JSFunction *func = mir->getCase(i);
847          LBlock *target = mir->getCaseBlock(i)->lir();
848
849  -       DebugOnly<bool> found = false;
850  +       DebugOnlyTor<bool> found = false;
851          for (size_t j = 0; j < propTable->numEntries(); j++) {
852              if (propTable->getFunction(j) != func)
853                  continue;
854  @@ -821,12 +821,12 @@ bool
855   CodeGenerator::visitReturn(LReturn *lir)
856   {
857   #if defined(JS_NUNBOX32)
858  -    DebugOnly<LAllocation *> type    = lir->getOperand(TYPE_INDEX);
859  -    DebugOnly<LAllocation *> payload = lir->getOperand(PAYLOAD_INDEX);
860  +    DebugOnlyTor<LAllocation *> type    = lir->getOperand(TYPE_INDEX);
861  +    DebugOnlyTor<LAllocation *> payload = lir->getOperand(PAYLOAD_INDEX);
862      JS_ASSERT(ToRegister(type)    == JSReturnReg_Type);
863      JS_ASSERT(ToRegister(payload) == JSReturnReg_Data);
864   #elif defined(JS_PUNBOX64)
865  -    DebugOnly<LAllocation *> result = lir->getOperand(0);
866  +    DebugOnlyTor<LAllocation *> result = lir->getOperand(0);
867      JS_ASSERT(ToRegister(result) == JSReturnReg);
868   #endif
869      // Don't emit a jump to the return label if this is the last block.
870  @@ -1317,7 +1317,7 @@ CodeGenerator::visitCallNative(LCallNative *call)
871      // Misc. temporary registers.
872      const Register tempReg = ToRegister(call->getTempReg());
873
874  -    DebugOnly<uint32_t> initialStack = masm.framePushed();
875  +    DebugOnlyTor<uint32_t> initialStack = masm.framePushed();
```

```
876
877       masm.checkStackAlignment();
878
879  @@ -1400,7 +1400,7 @@ CodeGenerator::visitCallDOMNative(LCallDOMNative *call)
880       const Register argPrivate   = ToRegister(call->getArgPrivate());
881       const Register argArgs      = ToRegister(call->getArgArgs());
882
883  -    DebugOnly<uint32_t> initialStack = masm.framePushed();
884  +    DebugOnlyTor<uint32_t> initialStack = masm.framePushed();
885
886       masm.checkStackAlignment();
887
888  @@ -2389,7 +2389,7 @@ CodeGenerator::maybeCreateScriptCounts()
889               MResumePoint *resume = block->entryResumePoint();
890               while (resume->caller())
891                   resume = resume->caller();
892  -            DebugOnly<uint32_t> offset = resume->pc() - script->code;
893  +            DebugOnlyTor<uint32_t> offset = resume->pc() - script->code;
894               JS_ASSERT(offset < script->length);
895           }
896
897  @@ -2694,7 +2694,7 @@ CodeGenerator::visitNewArray(LNewArray *lir)
898       JS_ASSERT(gen->info().executionMode() == SequentialExecution);
899       Register objReg = ToRegister(lir->output());
900       JSObject *templateObject = lir->mir()->templateObject();
901  -    DebugOnly<uint32_t> count = lir->mir()->count();
902  +    DebugOnlyTor<uint32_t> count = lir->mir()->count();
903
904       JS_ASSERT(count < JSObject::NELEMENTS_LIMIT);
905
906  @@ -3695,7 +3695,7 @@ CodeGenerator::visitIsNullOrLikeUndefined(
     LIsNullOrLikeUndefined *lir)
907       Register output = ToRegister(lir->output());
908
909       if (op == JSOP_EQ || op == JSOP_NE) {
910  -        MOZ_ASSERT(lir->mir()->lhs()->type() != MIRType_Object ||
911  +        TBB_MOZ_ASSERT(lir->mir()->lhs()->type() != MIRType_Object ||
912                      lir->mir()->operandMightEmulateUndefined(),
913                      "Operands which can't emulate undefined should have been folded")
                         ;
914
915  @@ -3783,7 +3783,7 @@ CodeGenerator::visitIsNullOrLikeUndefinedAndBranch(
     LIsNullOrLikeUndefinedAndBran
916               op = JSOP_EQ;
917           }
918
919  -        MOZ_ASSERT(lir->mir()->lhs()->type() != MIRType_Object ||
920  +        TBB_MOZ_ASSERT(lir->mir()->lhs()->type() != MIRType_Object ||
921                      lir->mir()->operandMightEmulateUndefined(),
922                      "Operands which can't emulate undefined should have been folded")
                         ;
```

```
923
924 @@ -3831,14 +3831,14 @@ static const VMFunction ConcatStringsInfo = FunctionInfo<
        ConcatStringsFn>(Concat
925  bool
926  CodeGenerator::visitEmulatesUndefined(LEmulatesUndefined *lir)
927  {
928 -    MOZ_ASSERT(lir->mir()->compareType() == MCompare::Compare_Undefined ||
929 +    TBB_MOZ_ASSERT(lir->mir()->compareType() == MCompare::Compare_Undefined ||
930                 lir->mir()->compareType() == MCompare::Compare_Null);
931 -    MOZ_ASSERT(lir->mir()->lhs()->type() == MIRType_Object);
932 -    MOZ_ASSERT(lir->mir()->operandMightEmulateUndefined(),
933 +    TBB_MOZ_ASSERT(lir->mir()->lhs()->type() == MIRType_Object);
934 +    TBB_MOZ_ASSERT(lir->mir()->operandMightEmulateUndefined(),
935                 "If the object couldn't emulate undefined, this should have been
                        folded.");
936
937      JSOp op = lir->mir()->jsop();
938 -    MOZ_ASSERT(op == JSOP_EQ || op == JSOP_NE, "Strict equality should have been
        folded");
939 +    TBB_MOZ_ASSERT(op == JSOP_EQ || op == JSOP_NE, "Strict equality should have been
        folded");
940
941      OutOfLineTestObjectWithLabels *ool = new OutOfLineTestObjectWithLabels();
942      if (!addOutOfLineCode(ool))
943 @@ -3866,13 +3866,13 @@ CodeGenerator::visitEmulatesUndefined(LEmulatesUndefined *lir
        )
944  bool
945  CodeGenerator::visitEmulatesUndefinedAndBranch(LEmulatesUndefinedAndBranch *lir)
946  {
947 -    MOZ_ASSERT(lir->mir()->compareType() == MCompare::Compare_Undefined ||
948 +    TBB_MOZ_ASSERT(lir->mir()->compareType() == MCompare::Compare_Undefined ||
949                 lir->mir()->compareType() == MCompare::Compare_Null);
950 -    MOZ_ASSERT(lir->mir()->operandMightEmulateUndefined(),
951 +    TBB_MOZ_ASSERT(lir->mir()->operandMightEmulateUndefined(),
952                 "Operands which can't emulate undefined should have been folded");
953
954      JSOp op = lir->mir()->jsop();
955 -    MOZ_ASSERT(op == JSOP_EQ || op == JSOP_NE, "Strict equality should have been
        folded");
956 +    TBB_MOZ_ASSERT(op == JSOP_EQ || op == JSOP_NE, "Strict equality should have been
        folded");
957
958      OutOfLineTestObject *ool = new OutOfLineTestObject();
959      if (!addOutOfLineCode(ool))
960 @@ -4136,7 +4136,7 @@ CodeGenerator::visitSetInitializedLength(LSetInitializedLength
        *lir)
961  bool
962  CodeGenerator::visitNotO(LNotO *lir)
963  {
964 -    MOZ_ASSERT(lir->mir()->operandMightEmulateUndefined(),
965 +    TBB_MOZ_ASSERT(lir->mir()->operandMightEmulateUndefined(),
```

```
                "This should be constant-folded if the object can't emulate undefined
                    .");

     OutOfLineTestObjectWithLabels *ool = new OutOfLineTestObjectWithLabels();
@@ -6585,7 +6585,7 @@ CodeGenerator::visitGetDOMProperty(LGetDOMProperty *ins)
     const Register PrivateReg = ToRegister(ins->getPrivReg());
     const Register ValueReg = ToRegister(ins->getValueReg());

-    DebugOnly<uint32_t> initialStack = masm.framePushed();
+    DebugOnlyTor<uint32_t> initialStack = masm.framePushed();

     masm.checkStackAlignment();

@@ -6654,7 +6654,7 @@ CodeGenerator::visitSetDOMProperty(LSetDOMProperty *ins)
     const Register PrivateReg = ToRegister(ins->getPrivReg());
     const Register ValueReg = ToRegister(ins->getValueReg());

-    DebugOnly<uint32_t> initialStack = masm.framePushed();
+    DebugOnlyTor<uint32_t> initialStack = masm.framePushed();

     masm.checkStackAlignment();

diff --git a/js/src/jit/InlineList.h b/js/src/jit/InlineList.h
index 441fdfe..37d2058 100644
--- a/js/src/jit/InlineList.h
+++ b/js/src/jit/InlineList.h
@@ -7,7 +7,7 @@
 #ifndef jit_InlineList_h
 #define jit_InlineList_h

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "jsutil.h"

@@ -40,7 +40,7 @@ class InlineForwardList : protected InlineForwardListNode<T>
     typedef InlineForwardListNode<T> Node;

     Node *tail_;
-    mozilla::DebugOnly<int> modifyCount_;
+    mozilla::DebugOnlyTor<int> modifyCount_;

     InlineForwardList<T> *thisFromConstructor() {
         return this;
@@ -140,7 +140,7 @@ private:
     InlineForwardListIterator<T>(const InlineForwardList<T> *owner)
       : prev(const_cast<Node *>(static_cast<const Node *>(owner))),
         iter(owner ? owner->next : NULL)
-#ifdef DEBUG
+#ifndef TOR_NASSERT
       , owner_(owner),
```

```
1016          modifyCount_(owner ? owner->modifyCount_.value : 0)
1017  #endif
1018 @@ -179,10 +179,10 @@ private:
1019      Node *prev;
1020      Node *iter;
1021
1022 -#ifdef DEBUG
1023 +#ifndef TOR_NASSERT
1024      const InlineForwardList<T> *owner_;
1025  #endif
1026 -    mozilla::DebugOnly<int> modifyCount_;
1027 +    mozilla::DebugOnlyTor<int> modifyCount_;
1028  };
1029
1030  template <typename T> class InlineList;
1031 diff --git a/js/src/jit/IonBuilder.cpp b/js/src/jit/IonBuilder.cpp
1032 index a0c70f5..6c4d8e3 100644
1033 --- a/js/src/jit/IonBuilder.cpp
1034 +++ b/js/src/jit/IonBuilder.cpp
1035 @@ -6,7 +6,7 @@
1036
1037  #include "jit/IonBuilder.h"
1038
1039 -#include "mozilla/DebugOnly.h"
1040 +#include "mozilla/DebugOnlyTor.h"
1041
1042  #include "builtin/Eval.h"
1043  #include "frontend/SourceNotes.h"
1044 @@ -31,7 +31,7 @@
1045  using namespace js;
1046  using namespace js::jit;
1047
1048 -using mozilla::DebugOnly;
1049 +using mozilla::DebugOnlyTor;
1050
1051  IonBuilder::IonBuilder(JSContext *cx, TempAllocator *temp, MIRGraph *graph,
1052                         BaselineInspector *inspector, CompileInfo *info,
                               BaselineFrame *baselineFrame,
1053 @@ -194,7 +194,7 @@ IonBuilder::getPolyCallTargets(types::StackTypeSet *calleeTypes,
1054              {
1055                  return false;
1056              }
1057 -            DebugOnly<bool> appendOk = targets.append(obj);
1058 +            DebugOnlyTor<bool> appendOk = targets.append(obj);
1059              JS_ASSERT(appendOk);
1060          } else {
1061              /* Temporarily disable heavyweight-function inlining. */
1062 @@ -209,7 +209,7 @@ IonBuilder::getPolyCallTargets(types::StackTypeSet *calleeTypes,
1063              }
1064              if (!typeObj->interpretedFunction->getOrCreateScript(cx))
1065                  return false;
```

```
-            DebugOnly<bool> appendOk = targets.append(typeObj->interpretedFunction);
+            DebugOnlyTor<bool> appendOk = targets.append(typeObj->
    interpretedFunction);
             JS_ASSERT(appendOk);

             *gotLambda = true;
@@ -2159,7 +2159,7 @@ IonBuilder::processBreak(JSOp op, jssrcnote *sn)

     // Find the break target.
     jsbytecode *target = pc + GetJumpOffset(pc);
-    DebugOnly<bool> found = false;
+    DebugOnlyTor<bool> found = false;

     if (SN_TYPE(sn) == SRC_BREAK2LABEL) {
         for (size_t i = labels_.length() - 1; i < labels_.length(); i--) {
@@ -2343,7 +2343,7 @@ IonBuilder::maybeLoop(JSOp op, jssrcnote *sn)
 void
 IonBuilder::assertValidLoopHeadOp(jsbytecode *pc)
 {
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     JS_ASSERT(JSOp(*pc) == JSOP_LOOPHEAD);

     // Make sure this is the next opcode after the loop header,
@@ -3772,7 +3772,7 @@ IonBuilder::makePolyInlineDispatch(JSContext *cx, CallInfo &
    callInfo,
         MResumePoint::New(current, pc, callerResumePoint_, MResumePoint::ResumeAt);
     if (!preCallResumePoint)
         return NULL;
-    DebugOnly<size_t> preCallFuncDefnIdx = preCallResumePoint->numOperands() - (((
    size_t) callInfo.argc()) + 2);
+    DebugOnlyTor<size_t> preCallFuncDefnIdx = preCallResumePoint->numOperands() -
    (((size_t) callInfo.argc()) + 2);
     JS_ASSERT(preCallResumePoint->getOperand(preCallFuncDefnIdx) == callInfo.fun());

     MDefinition *targetObject = getPropCache->object();
@@ -3816,7 +3816,7 @@ IonBuilder::makePolyInlineDispatch(JSContext *cx, CallInfo &
    callInfo,

     // The fallbackBlock inherits the state of the stack right before the getprop,
         which
     // means we have to pop off the target of the getprop before performing it.
-    DebugOnly<MDefinition *> checkTargetObject = fallbackBlock->pop();
+    DebugOnlyTor<MDefinition *> checkTargetObject = fallbackBlock->pop();
     JS_ASSERT(checkTargetObject == targetObject);

     // Remove the instructions leading to the function definition from the current
@@ -3994,7 +3994,7 @@ IonBuilder::inlineTypeObjectFallback(CallInfo &callInfo,
    MBasicBlock *dispatchBl
     if (!preCallResumePoint)
         return false;
```

```
1110
1111 -    DebugOnly<size_t> preCallFuncIndex = preCallResumePoint->numOperands() -
     callInfo.numFormals();
1112 +    DebugOnlyTor<size_t> preCallFuncIndex = preCallResumePoint->numOperands() -
     callInfo.numFormals();
1113      JS_ASSERT(preCallResumePoint->getOperand(preCallFuncIndex) == fallbackInfo.fun()
          );
1114
1115      // In the dispatch block, replace the function's slot entry with Undefined.
1116 @@ -4022,7 +4022,7 @@ IonBuilder::inlineTypeObjectFallback(CallInfo &callInfo,
     MBasicBlock *dispatchBl
1117
1118      // Since the getPropBlock inherited the stack from right before the
          MGetPropertyCache,
1119      // the target of the MGetPropertyCache is still on the stack.
1120 -    DebugOnly<MDefinition *> checkObject = getPropBlock->pop();
1121 +    DebugOnlyTor<MDefinition *> checkObject = getPropBlock->pop();
1122      JS_ASSERT(checkObject == cache->object());
1123
1124      // Move the MGetPropertyCache and friends into the getPropBlock.
1125 @@ -7387,7 +7387,7 @@ IonBuilder::TestCommonPropFunc(JSContext *cx, types::
     StackTypeSet *types, Handle
1126                  // above.
1127                  JS_ASSERT(propSet);
1128                  // Asking, freeze by asking.
1129 -                DebugOnly<bool> isOwn = propSet->isOwnProperty(cx, curType, false);
1130 +                DebugOnlyTor<bool> isOwn = propSet->isOwnProperty(cx, curType, false
     );
1131                  JS_ASSERT(!isOwn);
1132                  // Don't mark the proto. It will be held down by the shape
1133                  // guard. This allows us tp use properties found on prototypes
1134 diff --git a/js/src/jit/IonCaches.cpp b/js/src/jit/IonCaches.cpp
1135 index 933d42d..06f3ebb 100644
1136 --- a/js/src/jit/IonCaches.cpp
1137 +++ b/js/src/jit/IonCaches.cpp
1138 @@ -4,7 +4,7 @@
1139   * License, v. 2.0. If a copy of the MPL was not distributed with this
1140   * file, You can obtain one at http://mozilla.org/MPL/2.0/. */
1141
1142 -#include "mozilla/DebugOnly.h"
1143 +#include "mozilla/DebugOnlyTor.h"
1144
1145  #include "PerfSpewer.h"
1146  #include "CodeGenerator.h"
1147 @@ -23,7 +23,7 @@
1148  using namespace js;
1149  using namespace js::jit;
1150
1151 -using mozilla::DebugOnly;
1152 +using mozilla::DebugOnlyTor;
1153
```

```
 void
 CodeLocationJump::repoint(IonCode *code, MacroAssembler *masm)
@@ -893,7 +893,7 @@ GenerateCallGetter(JSContext *cx, IonScript *ion, MacroAssembler
    &masm,
      JS_ASSERT_IF(!callNative, IsCacheableGetPropCallPropertyOp(obj, holder, shape));

      // TODO: ensure stack is aligned?
-     DebugOnly<uint32_t> initialStack = masm.framePushed();
+     DebugOnlyTor<uint32_t> initialStack = masm.framePushed();

      Label success, exception;

@@ -1061,7 +1061,7 @@ GetPropertyIC::attachDOMProxyShadowed(JSContext *cx, IonScript
    *ion, JSObject *o
      // saveLive()
      masm.PushRegsInMask(liveRegs_);

-     DebugOnly<uint32_t> initialStack = masm.framePushed();
+     DebugOnlyTor<uint32_t> initialStack = masm.framePushed();

      // Remaining registers should be free, but we need to use |object| still
      // so leave it alone.
@@ -1848,7 +1848,7 @@ SetPropertyIC::attachSetterCall(JSContext *cx, IonScript *ion,
      Register argVpReg        = regSet.takeGeneral();

      // Ensure stack is aligned.
-     DebugOnly<uint32_t> initialStack = masm.framePushed();
+     DebugOnlyTor<uint32_t> initialStack = masm.framePushed();

      Label success, exception;

@@ -2282,7 +2282,7 @@ GetElementIC::attachTypedArrayElement(JSContext *cx, IonScript
    *ion, JSObject *o

      // The output register is not yet specialized as a float register, the only
      // way to accept float typed arrays for now is to return a Value type.
-     DebugOnly<bool> floatOutput = arrayType == TypedArray::TYPE_FLOAT32 ||
+     DebugOnlyTor<bool> floatOutput = arrayType == TypedArray::TYPE_FLOAT32 ||
                                    arrayType == TypedArray::TYPE_FLOAT64;
      JS_ASSERT_IF(!output().hasValue(), !floatOutput);

diff --git a/js/src/jit/IonFrames.h b/js/src/jit/IonFrames.h
index fcd33e6..33dfd94 100644
--- a/js/src/jit/IonFrames.h
+++ b/js/src/jit/IonFrames.h
@@ -9,7 +9,7 @@

 #ifdef JS_ION

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
```

```
 #include "jsfun.h"
 #include "jstypes.h"
@@ -123,7 +123,7 @@ class SafepointIndex
          uint32_t safepointOffset_;
     };

-    mozilla::DebugOnly<bool> resolved;
+    mozilla::DebugOnlyTor<bool> resolved;

   public:
     SafepointIndex(uint32_t displacement, LSafepoint *safepoint)
diff --git a/js/src/jit/LinearScan.cpp b/js/src/jit/LinearScan.cpp
index 1961da5..bf9be81 100644
--- a/js/src/jit/LinearScan.cpp
+++ b/js/src/jit/LinearScan.cpp
@@ -6,7 +6,7 @@

 #include <limits.h>

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "BitSet.h"
 #include "LinearScan.h"
@@ -17,7 +17,7 @@
 using namespace js;
 using namespace js::jit;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;

 /*
  * Merge virtual register intervals into the UnhandledQueue, taking advantage
@@ -476,7 +476,7 @@ LinearScanAllocator::populateSafepoints()
             // is not used with gcthings or nunboxes, or we would have to add the
                 input reg
             // to this safepoint.
             if (ins == reg->ins() && !reg->isTemp()) {
-                DebugOnly<LDefinition*> def = reg->def();
+                DebugOnlyTor<LDefinition*> def = reg->def();
                 JS_ASSERT_IF(def->policy() == LDefinition::MUST_REUSE_INPUT,
                              def->type() == LDefinition::GENERAL || def->type() ==
                                  LDefinition::DOUBLE);
                 continue;
diff --git a/js/src/jit/LiveRangeAllocator.cpp b/js/src/jit/LiveRangeAllocator.cpp
index e6d1eec..5f46a72 100644
--- a/js/src/jit/LiveRangeAllocator.cpp
+++ b/js/src/jit/LiveRangeAllocator.cpp
@@ -4,7 +4,7 @@
  * License, v. 2.0. If a copy of the MPL was not distributed with this
```

```
1251   * file, You can obtain one at http://mozilla.org/MPL/2.0/. */
1252
1253 -#include "mozilla/DebugOnly.h"
1254 +#include "mozilla/DebugOnlyTor.h"
1255
1256  #include "LiveRangeAllocator.h"
1257
1258 @@ -14,7 +14,7 @@
1259  using namespace js;
1260  using namespace js::jit;
1261
1262 -using mozilla::DebugOnly;
1263 +using mozilla::DebugOnlyTor;
1264
1265  int
1266  Requirement::priority() const
1267 @@ -355,7 +355,7 @@ VirtualRegister::getFirstInterval()
1268  template bool LiveRangeAllocator<LinearScanVirtualRegister>::buildLivenessInfo();
1269  template bool LiveRangeAllocator<BacktrackingVirtualRegister>::buildLivenessInfo();
1270
1271 -#ifdef DEBUG
1272 +#ifndef TOR_NASSERT
1273  static inline bool
1274  NextInstructionHasFixedUses(LBlock *block, LInstruction *ins)
1275  {
1276 @@ -642,8 +642,8 @@ LiveRangeAllocator<VREG>::buildLivenessInfo()
1277                  }
1278              }
1279
1280 -            DebugOnly<bool> hasUseRegister = false;
1281 -            DebugOnly<bool> hasUseRegisterAtStart = false;
1282 +            DebugOnlyTor<bool> hasUseRegister = false;
1283 +            DebugOnlyTor<bool> hasUseRegisterAtStart = false;
1284
1285              for (LInstruction::InputIterator alloc(**ins); alloc.more(); alloc.next
                     ()) {
1286                  if (alloc->isUse()) {
1287 diff --git a/js/src/jit/LiveRangeAllocator.h b/js/src/jit/LiveRangeAllocator.h
1288 index 4c349b1..f119eea 100644
1289 --- a/js/src/jit/LiveRangeAllocator.h
1290 +++ b/js/src/jit/LiveRangeAllocator.h
1291 @@ -7,7 +7,7 @@
1292  #ifndef jit_LiveRangeAllocator_h
1293  #define jit_LiveRangeAllocator_h
1294
1295 -#include "mozilla/DebugOnly.h"
1296 +#include "mozilla/DebugOnlyTor.h"
1297
1298  #include "RegisterAllocator.h"
1299  #include "StackSlotAllocator.h"
1300 @@ -122,7 +122,7 @@ UseCompatibleWith(const LUse *use, LAllocation alloc)
```

```
1301      return false;
1302  }
1303
1304 -#ifdef DEBUG
1305 +#ifndef TOR_NASSERT
1306
1307  static inline bool
1308  DefinitionCompatibleWith(LInstruction *ins, const LDefinition *def, LAllocation
          alloc)
1309 @@ -261,7 +261,7 @@ class LiveInterval
1310      const Range *getRange(size_t i) const {
1311          return &ranges_[i];
1312      }
1313 -    void setLastProcessedRange(size_t range, mozilla::DebugOnly<CodePosition> pos) {
1314 +    void setLastProcessedRange(size_t range, mozilla::DebugOnlyTor<CodePosition> pos
         ) {
1315          // If the range starts after pos, we may not be able to use
1316          // it in the next lastProcessedRangeIfValid call.
1317          JS_ASSERT(ranges_[range].from <= pos);
1318 diff --git a/js/src/jit/Lowering.cpp b/js/src/jit/Lowering.cpp
1319 index fd1dc57..9ee6072 100644
1320 --- a/js/src/jit/Lowering.cpp
1321 +++ b/js/src/jit/Lowering.cpp
1322 @@ -14,7 +14,7 @@
1323  #include "jsbool.h"
1324  #include "jsnum.h"
1325  #include "shared/Lowering-shared-inl.h"
1326 -#include "mozilla/DebugOnly.h"
1327 +#include "mozilla/DebugOnlyTor.h"
1328
1329  using namespace js;
1330  using namespace jit;
1331 @@ -263,7 +263,7 @@ LIRGenerator::visitPrepareCall(MPrepareCall *ins)
1332  {
1333      allocateArguments(ins->argc());
1334
1335 -#ifdef DEBUG
1336 +#ifndef TOR_NASSERT
1337      if (!prepareCallStack_.append(ins))
1338          return false;
1339  #endif
1340 @@ -380,7 +380,7 @@ LIRGenerator::visitCall(MCall *call)
1341          GetTempRegForIntArg(0, 0, &cxReg);
1342          GetTempRegForIntArg(1, 0, &objReg);
1343          GetTempRegForIntArg(2, 0, &privReg);
1344 -        mozilla::DebugOnly<bool> ok = GetTempRegForIntArg(3, 0, &argsReg);
1345 +        mozilla::DebugOnlyTor<bool> ok = GetTempRegForIntArg(3, 0, &argsReg);
1346          MOZ_ASSERT(ok, "How can we not have four temp registers?");
1347          LCallDOMNative *lir = new LCallDOMNative(argslot, tempFixed(cxReg),
1348                                                   tempFixed(objReg), tempFixed(
                                                       privReg),
```

```
@@ -398,7 +398,7 @@ LIRGenerator::visitCall(MCall *call)

             // Even though this is just a temp reg, use the same API to avoid
             // register collisions.
-            mozilla::DebugOnly<bool> ok = GetTempRegForIntArg(3, 0, &tmpReg);
+            mozilla::DebugOnlyTor<bool> ok = GetTempRegForIntArg(3, 0, &tmpReg);
             MOZ_ASSERT(ok, "How can we not have four temp registers?");

             LCallNative *lir = new LCallNative(argslot, tempFixed(cxReg),
@@ -1395,7 +1395,7 @@ bool
 LIRGenerator::visitToDouble(MToDouble *convert)
 {
     MDefinition *opd = convert->input();
-    mozilla::DebugOnly<MToDouble::ConversionKind> conversion = convert->conversion()
    ;
+    mozilla::DebugOnlyTor<MToDouble::ConversionKind> conversion = convert->
    conversion();

     switch (opd->type()) {
       case MIRType_Value:
@@ -2767,7 +2767,7 @@ LIRGenerator::visitSetDOMProperty(MSetDOMProperty *ins)
     // don't clobber registers we're already using.
     Register tempReg1, tempReg2;
     GetTempRegForIntArg(4, 0, &tempReg1);
-    mozilla::DebugOnly<bool> ok = GetTempRegForIntArg(5, 0, &tempReg2);
+    mozilla::DebugOnlyTor<bool> ok = GetTempRegForIntArg(5, 0, &tempReg2);
     MOZ_ASSERT(ok, "How can we not have six temp registers?");
     if (!useBoxFixed(lir, LSetDOMProperty::Value, val, tempReg1, tempReg2))
         return false;
@@ -2782,7 +2782,7 @@ LIRGenerator::visitGetDOMProperty(MGetDOMProperty *ins)
     GetTempRegForIntArg(0, 0, &cxReg);
     GetTempRegForIntArg(1, 0, &objReg);
     GetTempRegForIntArg(2, 0, &privReg);
-    mozilla::DebugOnly<bool> ok = GetTempRegForIntArg(3, 0, &valueReg);
+    mozilla::DebugOnlyTor<bool> ok = GetTempRegForIntArg(3, 0, &valueReg);
     MOZ_ASSERT(ok, "How can we not have four temp registers?");
     LGetDOMProperty *lir = new LGetDOMProperty(tempFixed(cxReg),
                                                useFixed(ins->object(), objReg),
diff --git a/js/src/jit/Lowering.h b/js/src/jit/Lowering.h
index 3d67a2d..edb9d9a 100644
--- a/js/src/jit/Lowering.h
+++ b/js/src/jit/Lowering.h
@@ -37,7 +37,7 @@ class LIRGenerator : public LIRGeneratorSpecific
     // The maximum depth, for framesizeclass determination.
     uint32_t maxargslots_;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     // In debug builds, check MPrepareCall and MCall are properly
     // nested. The argslots_ mechanism relies on this.
     Vector<MPrepareCall *, 4, SystemAllocPolicy> prepareCallStack_;
```

```
diff --git a/js/src/jit/MIR.cpp b/js/src/jit/MIR.cpp
index eea62ff..0c8da1d 100644
--- a/js/src/jit/MIR.cpp
+++ b/js/src/jit/MIR.cpp
@@ -644,7 +644,7 @@ MPhi::reserveLength(size_t length)
     // capacity. This permits use of addInput() instead of addInputSlow(), the
     // latter of which may call realloc().
     JS_ASSERT(numOperands() == 0);
-#if DEBUG
+#if !TOR_NASSERT
     capacity_ = length;
 #endif
     return inputs_.reserve(length);
@@ -691,7 +691,7 @@ jit::MergeTypes(MIRType *ptype, types::StackTypeSet **ptypeSet,
 void
 MPhi::specializeType()
 {
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     JS_ASSERT(!specialized_);
     specialized_ = true;
 #endif
diff --git a/js/src/jit/MIR.h b/js/src/jit/MIR.h
index e9bc029..6d6a68a 100644
--- a/js/src/jit/MIR.h
+++ b/js/src/jit/MIR.h
@@ -483,7 +483,7 @@ class MDefinition : public MNode

     void setVirtualRegister(uint32_t vreg) {
         virtualRegister_ = vreg;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
         setLoweredUnchecked();
 #endif
     }
@@ -3601,7 +3601,7 @@ class MPhi : public MDefinition, public InlineForwardListNode<
     MPhi>
     bool triedToSpecialize_;
     bool isIterator_;

-#if DEBUG
+#ifndef TOR_NASSERT
     bool specialized_;
     uint32_t capacity_;
 #endif
@@ -3611,7 +3611,7 @@ class MPhi : public MDefinition, public InlineForwardListNode<
     MPhi>
         hasBackedgeType_(false),
         triedToSpecialize_(false),
         isIterator_(false)
-#if DEBUG
```

```
+#ifndef TOR_NASSERT
         , specialized_(false)
         , capacity_(0)
 #endif
diff --git a/js/src/jit/arm/Assembler-arm.cpp b/js/src/jit/arm/Assembler-arm.cpp
index 57a3aa2..e47c3d3 100644
--- a/js/src/jit/arm/Assembler-arm.cpp
+++ b/js/src/jit/arm/Assembler-arm.cpp
@@ -4,7 +4,7 @@
  * License, v. 2.0. If a copy of the MPL was not distributed with this
  * file, You can obtain one at http://mozilla.org/MPL/2.0/. */

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "Assembler-arm.h"
 #include "MacroAssembler-arm.h"
@@ -2312,7 +2312,7 @@ Assembler::retarget(Label *label, Label *target)
         } else {
             // The target is unbound and unused.  We can just take the head of
             // the list hanging off of label, and dump that into target.
-            DebugOnly<uint32_t> prev = target->use(label->offset());
+            DebugOnlyTor<uint32_t> prev = target->use(label->offset());
             JS_ASSERT((int32_t)prev == Label::INVALID_OFFSET);
         }
     }
@@ -2651,7 +2651,7 @@ Assembler::ToggleToJmp(CodeLocationLabel inst_)
     uint32_t *ptr = (uint32_t *)inst_.raw();

-    DebugOnly<Instruction *> inst = (Instruction *)inst_.raw();
+    DebugOnlyTor<Instruction *> inst = (Instruction *)inst_.raw();
     JS_ASSERT(inst->is<InstCMP>());

     // Zero bits 20-27, then set 24-27 to be correct for a branch.
@@ -2665,7 +2665,7 @@ Assembler::ToggleToCmp(CodeLocationLabel inst_)
     uint32_t *ptr = (uint32_t *)inst_.raw();

-    DebugOnly<Instruction *> inst = (Instruction *)inst_.raw();
+    DebugOnlyTor<Instruction *> inst = (Instruction *)inst_.raw();
     JS_ASSERT(inst->is<InstBImm>());

     // Ensure that this masking operation doesn't affect the offset of the
diff --git a/js/src/jit/arm/MacroAssembler-arm.cpp b/js/src/jit/arm/MacroAssembler-
    arm.cpp
index b7a3167..a030f54 100644
--- a/js/src/jit/arm/MacroAssembler-arm.cpp
+++ b/js/src/jit/arm/MacroAssembler-arm.cpp
@@ -4,7 +4,7 @@
  * License, v. 2.0. If a copy of the MPL was not distributed with this
```

```
1497   * file, You can obtain one at http://mozilla.org/MPL/2.0/. */
1498
1499 -#include "mozilla/DebugOnly.h"
1500 +#include "mozilla/DebugOnlyTor.h"
1501  #include "mozilla/MathAlgorithms.h"
1502
1503  #include "jit/arm/MacroAssembler-arm.h"
1504 @@ -930,7 +930,7 @@ MacroAssemblerARM::ma_str(Register rt, const Operand &addr, Index
       mode, Conditio
1505       ma_dtr(IsStore, rt, addr, mode, cc);
1506  }
1507  void
1508 -MacroAssemblerARM::ma_strd(Register rt, DebugOnly<Register> rt2, EDtrAddr addr,
      Index mode, Condition cc)
1509 +MacroAssemblerARM::ma_strd(Register rt, DebugOnlyTor<Register> rt2, EDtrAddr addr,
      Index mode, Condition cc)
1510  {
1511       JS_ASSERT((rt.code() & 1) == 0);
1512       JS_ASSERT(rt2.value.code() == rt.code() + 1);
1513 @@ -971,7 +971,7 @@ MacroAssemblerARM::ma_ldrsb(EDtrAddr addr, Register rt, Index
      mode, Condition cc
1514       as_extdtr(IsLoad, 8, true, mode, rt, addr, cc);
1515  }
1516  void
1517 -MacroAssemblerARM::ma_ldrd(EDtrAddr addr, Register rt, DebugOnly<Register> rt2,
1518 +MacroAssemblerARM::ma_ldrd(EDtrAddr addr, Register rt, DebugOnlyTor<Register> rt2,
1519                            Index mode, Condition cc)
1520  {
1521       JS_ASSERT((rt.code() & 1) == 0);
1522 @@ -1466,13 +1466,13 @@ MacroAssemblerARM::ma_vstr(VFPRegister src, Register base,
      Register index, int32
1523  bool
1524  MacroAssemblerARMCompat::buildFakeExitFrame(const Register &scratch, uint32_t *
       offset)
1525  {
1526 -    DebugOnly<uint32_t> initialDepth = framePushed();
1527 +    DebugOnlyTor<uint32_t> initialDepth = framePushed();
1528       uint32_t descriptor = MakeFrameDescriptor(framePushed(), IonFrame_OptimizedJS);
1529
1530       Push(Imm32(descriptor)); // descriptor_
1531
1532       enterNoPool();
1533 -    DebugOnly<uint32_t> offsetBeforePush = currentOffset();
1534 +    DebugOnlyTor<uint32_t> offsetBeforePush = currentOffset();
1535       Push(pc); // actually pushes $pc + 8.
1536
1537       // Consume an additional 4 bytes. The start of the next instruction will
1538 @@ -1492,7 +1492,7 @@ MacroAssemblerARMCompat::buildFakeExitFrame(const Register &
      scratch, uint32_t *o
1539  bool
1540  MacroAssemblerARMCompat::buildOOLFakeExitFrame(void *fakeReturnAddr)
```

```
 {
-    DebugOnly<uint32_t> initialDepth = framePushed();
+    DebugOnlyTor<uint32_t> initialDepth = framePushed();
     uint32_t descriptor = MakeFrameDescriptor(framePushed(), IonFrame_OptimizedJS);

     Push(Imm32(descriptor)); // descriptor_
diff --git a/js/src/jit/arm/MacroAssembler-arm.h b/js/src/jit/arm/MacroAssembler-arm.h
index 04d68af..1b37eb8 100644
--- a/js/src/jit/arm/MacroAssembler-arm.h
+++ b/js/src/jit/arm/MacroAssembler-arm.h
@@ -7,7 +7,7 @@
 #ifndef jit_arm_MacroAssembler_arm_h
 #define jit_arm_MacroAssembler_arm_h

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "jit/arm/Assembler-arm.h"
 #include "jit/IonCaches.h"
@@ -15,7 +15,7 @@
 #include "jit/MoveResolver.h"
 #include "jsopcode.h"

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;

 namespace js {
 namespace jit {
@@ -258,10 +258,10 @@ class MacroAssemblerARM : public Assembler
     void ma_ldrh(EDtrAddr addr, Register rt, Index mode = Offset, Condition cc =
         Always);
     void ma_ldrsh(EDtrAddr addr, Register rt, Index mode = Offset, Condition cc =
         Always);
     void ma_ldrsb(EDtrAddr addr, Register rt, Index mode = Offset, Condition cc =
         Always);
-    void ma_ldrd(EDtrAddr addr, Register rt, DebugOnly<Register> rt2, Index mode =
    Offset, Condition cc = Always);
+    void ma_ldrd(EDtrAddr addr, Register rt, DebugOnlyTor<Register> rt2, Index mode
    = Offset, Condition cc = Always);
     void ma_strb(Register rt, DTRAddr addr, Index mode = Offset, Condition cc =
         Always);
     void ma_strh(Register rt, EDtrAddr addr, Index mode = Offset, Condition cc =
         Always);
-    void ma_strd(Register rt, DebugOnly<Register> rt2, EDtrAddr addr, Index mode =
    Offset, Condition cc = Always);
+    void ma_strd(Register rt, DebugOnlyTor<Register> rt2, EDtrAddr addr, Index mode
    = Offset, Condition cc = Always);
     // specialty for moving N bits of data, where n == 8,16,32,64
     BufferOffset ma_dataTransferN(LoadStore ls, int size, bool IsSigned,
                                   Register rn, Register rm, Register rt,
```

```
diff --git a/js/src/jit/shared/Assembler-shared.h b/js/src/jit/shared/Assembler-shared.h
index fc253d8..e3ec5ec6 100644
--- a/js/src/jit/shared/Assembler-shared.h
+++ b/js/src/jit/shared/Assembler-shared.h
@@ -9,7 +9,7 @@

 #include <limits.h>

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/PodOperations.h"

 #include "jit/IonAllocPolicy.h"
@@ -205,7 +205,7 @@ struct LabelBase
     void operator =(const LabelBase &label);
     static int id_count;
   public:
-    mozilla::DebugOnly <int> id;
+    mozilla::DebugOnlyTor <int> id;
     static const int32_t INVALID_OFFSET = -1;

     LabelBase() : offset_(INVALID_OFFSET), bound_(false), id(id_count++)
@@ -434,7 +434,7 @@ class CodeOffsetLabel
 class CodeLocationJump
 {
     uint8_t *raw_;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     bool absolute_;
     void setAbsolute() {
         absolute_ = true;
@@ -500,7 +500,7 @@ class CodeLocationJump
 class CodeLocationLabel
 {
     uint8_t *raw_;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     bool absolute_;
     void setAbsolute() {
         absolute_ = true;
diff --git a/js/src/jit/shared/CodeGenerator-x86-shared.cpp b/js/src/jit/shared/CodeGenerator-x86-shared.cpp
index 363ce8a..87e7e81 100644
--- a/js/src/jit/shared/CodeGenerator-x86-shared.cpp
+++ b/js/src/jit/shared/CodeGenerator-x86-shared.cpp
@@ -4,7 +4,7 @@
  * License, v. 2.0. If a copy of the MPL was not distributed with this
  * file, You can obtain one at http://mozilla.org/MPL/2.0/. */

-#include "mozilla/DebugOnly.h"
```

```
+#include "mozilla/DebugOnlyTor.h"

 #include "jscntxt.h"
 #include "jscompartment.h"
@@ -519,7 +519,7 @@ CodeGeneratorX86Shared::visitOutOfLineUndoALUOperation(
     OutOfLineUndoALUOperation
      LInstruction *ins = ool->ins();
      Register reg = ToRegister(ins->getDef(0));

-     mozilla::DebugOnly<LAllocation *> lhs = ins->getOperand(0);
+     mozilla::DebugOnlyTor<LAllocation *> lhs = ins->getOperand(0);
      LAllocation *rhs = ins->getOperand(1);

      JS_ASSERT(reg == ToRegister(lhs));
@@ -684,7 +684,7 @@ CodeGeneratorX86Shared::visitDivPowTwoI(LDivPowTwoI *ins)
 {
      Register lhs = ToRegister(ins->numerator());
      Register lhsCopy = ToRegister(ins->numeratorCopy());
-     mozilla::DebugOnly<Register> output = ToRegister(ins->output());
+     mozilla::DebugOnlyTor<Register> output = ToRegister(ins->output());
      int32_t shift = ins->shift();

      // We use defineReuseInput so these should always be the same, which is
diff --git a/js/src/jit/shared/MacroAssembler-x86-shared.h b/js/src/jit/shared/
    MacroAssembler-x86-shared.h
index 6d537f8..8ef0794 100644
--- a/js/src/jit/shared/MacroAssembler-x86-shared.h
+++ b/js/src/jit/shared/MacroAssembler-x86-shared.h
@@ -7,7 +7,7 @@
 #ifndef jit_shared_MacroAssembler_x86_shared_h
 #define jit_shared_MacroAssembler_x86_shared_h

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #ifdef JS_CPU_X86
 # include "jit/x86/Assembler-x86.h"
@@ -455,7 +455,7 @@ class MacroAssemblerX86Shared : public Assembler
      // Builds an exit frame on the stack, with a return address to an internal
      // non-function. Returns offset to be passed to markSafepointAt().
      bool buildFakeExitFrame(const Register &scratch, uint32_t *offset) {
-         mozilla::DebugOnly<uint32_t> initialDepth = framePushed();
+         mozilla::DebugOnlyTor<uint32_t> initialDepth = framePushed();

          CodeLabel cl;
          mov(cl.dest(), scratch);
diff --git a/js/src/jit/x64/Assembler-x64.cpp b/js/src/jit/x64/Assembler-x64.cpp
index e4f253b..3b641f3 100644
--- a/js/src/jit/x64/Assembler-x64.cpp
+++ b/js/src/jit/x64/Assembler-x64.cpp
@@ -158,7 +158,7 @@ Assembler::finish()
```

```
1680
1681      // Zero the extended jumps table.
1682      for (size_t i = 0; i < jumps_.length(); i++) {
1683 -#ifdef DEBUG
1684 +#ifndef TOR_NASSERT
1685          size_t oldSize = masm.size();
1686  #endif
1687          masm.jmp_rip(0);
1688 diff --git a/js/src/jit/x86/CodeGenerator-x86.cpp b/js/src/jit/x86/CodeGenerator-x86.
     cpp
1689 index bc4f736..7f93a89 100644
1690 --- a/js/src/jit/x86/CodeGenerator-x86.cpp
1691 +++ b/js/src/jit/x86/CodeGenerator-x86.cpp
1692 @@ -4,7 +4,7 @@
1693   * License, v. 2.0. If a copy of the MPL was not distributed with this
1694   * file, You can obtain one at http://mozilla.org/MPL/2.0/. */
1695
1696 -#include "mozilla/DebugOnly.h"
1697 +#include "mozilla/DebugOnlyTor.h"
1698
1699  #include "jsnum.h"
1700
1701 @@ -20,7 +20,7 @@
1702  using namespace js;
1703  using namespace js::jit;
1704
1705 -using mozilla::DebugOnly;
1706 +using mozilla::DebugOnlyTor;
1707  using mozilla::DoubleExponentBias;
1708  using mozilla::DoubleExponentShift;
1709
1710 @@ -105,7 +105,7 @@ CodeGeneratorX86::visitBox(LBox *box)
1711  {
1712      const LDefinition *type = box->getDef(TYPE_INDEX);
1713
1714 -    DebugOnly<const LAllocation *> a = box->getOperand(0);
1715 +    DebugOnlyTor<const LAllocation *> a = box->getOperand(0);
1716      JS_ASSERT(!a->isConstant());
1717
1718      // On x86, the input operand and the output payload have the same
1719 diff --git a/js/src/jsanalyze.cpp b/js/src/jsanalyze.cpp
1720 index b42dd4b..b123334 100644
1721 --- a/js/src/jsanalyze.cpp
1722 +++ b/js/src/jsanalyze.cpp
1723 @@ -6,7 +6,7 @@
1724
1725  #include "jsanalyze.h"
1726
1727 -#include "mozilla/DebugOnly.h"
1728 +#include "mozilla/DebugOnlyTor.h"
1729  #include "mozilla/PodOperations.h"
```

```
 #include "jscompartment.h"
@@ -19,7 +19,7 @@
 using namespace js;
 using namespace js::analyze;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;
 using mozilla::PodCopy;
 using mozilla::PodZero;

@@ -655,7 +655,7 @@ ScriptAnalysis::analyzeLifetimes(JSContext *cx)
             loop->lastBlock = offset;

         if (code->exceptionEntry) {
-            DebugOnly<bool> found = false;
+            DebugOnlyTor<bool> found = false;
             JSTryNote *tn = script_->trynotes()->vector;
             JSTryNote *tnlimit = tn + script_->trynotes()->length;
             for (; tn < tnlimit; tn++) {
diff --git a/js/src/jsapi.cpp b/js/src/jsapi.cpp
index 3632a74..b91f07c 100644
--- a/js/src/jsapi.cpp
+++ b/js/src/jsapi.cpp
@@ -1059,7 +1059,7 @@ JSRuntime::abortIfWrongThread() const
         MOZ_CRASH();
 }

-#ifdef DEBUG
+#ifndef TOR_NASSERT
 JS_FRIEND_API(void)
 JSRuntime::assertValidThread() const
 {
diff --git a/js/src/jsarray.cpp b/js/src/jsarray.cpp
index 12bb291..90dccd6 100644
--- a/js/src/jsarray.cpp
+++ b/js/src/jsarray.cpp
@@ -6,7 +6,7 @@

 #include "jsarray.h"

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/FloatingPoint.h"
 #include "mozilla/MathAlgorithms.h"
 #include "mozilla/Util.h"
@@ -43,7 +43,7 @@ using namespace js::types;

 using mozilla::Abs;
 using mozilla::ArrayLength;
-using mozilla::DebugOnly;
```

```
+using mozilla::DebugOnlyTor;
 using mozilla::IsNaN;
 using mozilla::PointerRangeSize;

@@ -2851,7 +2851,7 @@ EnsureNewArrayElements(JSContext *cx, JSObject *obj, uint32_t
    length)
       * If ensureElements creates dynamically allocated slots, then having
       * fixedSlots is a waste.
       */
-    DebugOnly<uint32_t> cap = obj->getDenseCapacity();
+    DebugOnlyTor<uint32_t> cap = obj->getDenseCapacity();

     if (!obj->ensureElements(cx, length))
         return false;
diff --git a/js/src/jsboolinlines.h b/js/src/jsboolinlines.h
index b85d7ea..c622ac9 100644
--- a/js/src/jsboolinlines.h
+++ b/js/src/jsboolinlines.h
@@ -7,7 +7,7 @@
 #ifndef jsboolinlines_h
 #define jsboolinlines_h

-#include "mozilla/Assertions.h"
+#include "mozilla/AssertionsTor.h"
 #include "mozilla/Likely.h"

 #include "js/RootingAPI.h"
@@ -33,7 +33,7 @@ EmulatesUndefined(JSObject *obj)
 {
     JSObject *actual = MOZ_LIKELY(!obj->isWrapper()) ? obj : UncheckedUnwrap(obj);
     bool emulatesUndefined = actual->getClass()->emulatesUndefined();
-    MOZ_ASSERT_IF(emulatesUndefined, obj->type()->flags & types::
    OBJECT_FLAG_EMULATES_UNDEFINED);
+    TBB_MOZ_ASSERT_IF(emulatesUndefined, obj->type()->flags & types::
    OBJECT_FLAG_EMULATES_UNDEFINED);
     return emulatesUndefined;
 }

diff --git a/js/src/jscntxt.cpp b/js/src/jscntxt.cpp
index 9e16009f..8e6bd31 100644
--- a/js/src/jscntxt.cpp
+++ b/js/src/jscntxt.cpp
@@ -14,7 +14,7 @@
 #include <stdarg.h>
 #include <string.h>

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #ifdef ANDROID
 # include <android/log.h>
```

```
@@ -56,7 +56,7 @@
 using namespace js;
 using namespace js::gc;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;
 using mozilla::PodArrayZero;
 using mozilla::PodZero;
 using mozilla::PointerRangeSize;
@@ -616,7 +616,7 @@ js::ReportUsageError(JSContext *cx, HandleObject callee, const
    char *msg)
     const char *usageStr = "usage";
     PropertyName *usageAtom = Atomize(cx, usageStr, strlen(usageStr))->
         asPropertyName();
     RootedId id(cx, NameToId(usageAtom));
-    DebugOnly<Shape *> shape = static_cast<Shape *>(callee->nativeLookup(cx, id));
+    DebugOnlyTor<Shape *> shape = static_cast<Shape *>(callee->nativeLookup(cx, id))
    ;
     JS_ASSERT(!shape->configurable());
     JS_ASSERT(!shape->writable());
     JS_ASSERT(shape->hasDefaultGetter());
diff --git a/js/src/jscntxt.h b/js/src/jscntxt.h
index b7aa4b8..8c992c9 100644
--- a/js/src/jscntxt.h
+++ b/js/src/jscntxt.h
@@ -676,7 +676,7 @@ struct JSRuntime : public JS::shadow::Runtime,
      * Protects all data that is touched in this process.
      */
     PRLock *operationCallbackLock;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     PRThread *operationCallbackOwner;
 #endif
   public:
@@ -689,13 +689,13 @@ struct JSRuntime : public JS::shadow::Runtime,
         AutoLockForOperationCallback(JSRuntime *rt MOZ_GUARD_OBJECT_NOTIFIER_PARAM)
             : rt(rt) {
             MOZ_GUARD_OBJECT_NOTIFIER_INIT;
             PR_Lock(rt->operationCallbackLock);
-#ifdef DEBUG
+#ifndef TOR_NASSERT
             rt->operationCallbackOwner = PR_GetCurrentThread();
 #endif
         }
         ~AutoLockForOperationCallback() {
             JS_ASSERT(rt->operationCallbackOwner == PR_GetCurrentThread());
-#ifdef DEBUG
+#ifndef TOR_NASSERT
             rt->operationCallbackOwner = NULL;
 #endif
             PR_Unlock(rt->operationCallbackLock);
```

```
@@ -711,7 +711,7 @@ struct JSRuntime : public JS::shadow::Runtime,
     };

     bool currentThreadOwnsOperationCallbackLock() {
-#if defined(JS_THREADSAFE) && defined(DEBUG)
+#if defined(JS_THREADSAFE) && !defined(TOR_NASSERT)
         return operationCallbackOwner == PR_GetCurrentThread();
 #else
         return true;
@@ -746,7 +746,7 @@ struct JSRuntime : public JS::shadow::Runtime,
     void clearOwnerThread();
     void setOwnerThread();
     JS_FRIEND_API(void) abortIfWrongThread() const;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     JS_FRIEND_API(void) assertValidThread() const;
 #else
     void assertValidThread() const {}
@@ -893,7 +893,7 @@ struct JSRuntime : public JS::shadow::Runtime,
     /* The request depth for this thread. */
     unsigned            requestDepth;

-# ifdef DEBUG
+#ifndef TOR_NASSERT
     unsigned            checkRequestDepth;
 # endif
 #endif
@@ -989,7 +989,7 @@ struct JSRuntime : public JS::shadow::Runtime,
      */
     bool                gcStrictCompartmentChecking;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     /*
      * If this is 0, all cross-compartment proxies must be registered in the
      * wrapper map. This checking must be disabled temporarily while creating
@@ -1037,7 +1037,7 @@ struct JSRuntime : public JS::shadow::Runtime,
      */
     js::gc::ArenaHeader *gcArenasAllocatedDuringSweep;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     js::gc::MarkingValidator *gcMarkingValidator;
 #endif

@@ -1367,7 +1367,7 @@ struct JSRuntime : public JS::shadow::Runtime,

     js::ScriptDataTable scriptDataTable;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
```

```
1927      size_t              noGCOrAllocationCheck;
1928  #endif
1929
1930 @@ -1505,7 +1505,7 @@ struct JSRuntime : public JS::shadow::Runtime,
1931  #endif
1932      }
1933
1934 -#ifdef DEBUG
1935 +#ifndef TOR_NASSERT
1936    public:
1937      js::AutoEnterPolicy *enteredPolicy;
1938  #endif
1939 @@ -1718,7 +1718,7 @@ struct JSContext : js::ThreadSafeContext,
1940      bool hasEnteredCompartment() const {
1941          return enterCompartmentDepth_ > 0;
1942      }
1943 -#ifdef DEBUG
1944 +#ifndef TOR_NASSERT
1945      unsigned getEnterCompartmentDepth() const {
1946          return enterCompartmentDepth_;
1947      }
1948 @@ -1906,7 +1906,7 @@ struct JSContext : js::ThreadSafeContext,
1949
1950      JSAtomState & names() { return runtime()->atomState; }
1951
1952 -#ifdef DEBUG
1953 +#ifndef TOR_NASSERT
1954      /*
1955       * Controls whether a quadratic-complexity assertion is performed during
1956       * stack iteration; defaults to true.
1957 @@ -2420,14 +2420,14 @@ class AutoObjectHashSet : public AutoHashSetRooter<JSObject
         *>
1958
1959  class AutoAssertNoException
1960  {
1961 -#ifdef DEBUG
1962 +#ifndef TOR_NASSERT
1963      JSContext *cx;
1964      bool hadException;
1965  #endif
1966
1967    public:
1968      AutoAssertNoException(JSContext *cx)
1969 -#ifdef DEBUG
1970 +#ifndef TOR_NASSERT
1971        : cx(cx),
1972          hadException(cx->isExceptionPending())
1973  #endif
1974 @@ -2497,7 +2497,7 @@ JSBool intrinsic_HaveSameClass(JSContext *cx, unsigned argc,
         Value *vp);
1975  JSBool intrinsic_ShouldForceSequential(JSContext *cx, unsigned argc, Value *vp);
```

```
 JSBool intrinsic_NewParallelArray(JSContext *cx, unsigned argc, Value *vp);

-#ifdef DEBUG
+#ifndef TOR_NASSERT
 JSBool intrinsic_Dump(JSContext *cx, unsigned argc, Value *vp);
 #endif

diff --git a/js/src/jscntxtinlines.h b/js/src/jscntxtinlines.h
index 2838b60..b09ed88 100644
--- a/js/src/jscntxtinlines.h
+++ b/js/src/jscntxtinlines.h
@@ -314,7 +314,7 @@ CallJSNative(JSContext *cx, Native native, const CallArgs &args)
 {
     JS_CHECK_RECURSION(cx, return false);

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     bool alreadyThrowing = cx->isExceptionPending();
 #endif
     assertSameCompartment(cx, args);
@@ -330,7 +330,7 @@ STATIC_PRECONDITION_ASSUME(ubound(args.argv_) >= argc)
 JS_ALWAYS_INLINE bool
 CallNativeImpl(JSContext *cx, NativeImpl impl, const CallArgs &args)
 {
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     bool alreadyThrowing = cx->isExceptionPending();
 #endif
     assertSameCompartment(cx, args);
@@ -346,7 +346,7 @@ STATIC_PRECONDITION(ubound(args.argv_) >= argc)
 JS_ALWAYS_INLINE bool
 CallJSNativeConstructor(JSContext *cx, Native native, const CallArgs &args)
 {
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     RootedObject callee(cx, &args.callee());
 #endif

diff --git a/js/src/jscompartment.cpp b/js/src/jscompartment.cpp
index c448e10..1a668ef 100644
--- a/js/src/jscompartment.cpp
+++ b/js/src/jscompartment.cpp
@@ -6,7 +6,7 @@

 #include "jscompartment.h"

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "jscntxt.h"
 #include "jsgc.h"
```

```
@@ -30,7 +30,7 @@
 using namespace js;
 using namespace js::gc;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;

 JSCompartment::JSCompartment(Zone *zone, const JS::CompartmentOptions &options = JS
     ::CompartmentOptions())
   : zone_(zone),
@@ -270,7 +270,7 @@ JSCompartment::wrap(JSContext *cx, MutableHandleValue vp,
    HandleObject existingA
     if (WrapperMap::Ptr p = crossCompartmentWrappers.lookup(key)) {
         vp.set(p->value);
         if (vp.isObject()) {
-            DebugOnly<JSObject *> obj = &vp.toObject();
+            DebugOnlyTor<JSObject *> obj = &vp.toObject();
             JS_ASSERT(obj->isCrossCompartmentWrapper());
             JS_ASSERT(obj->getParent() == global);
         }
diff --git a/js/src/jsgc.cpp b/js/src/jsgc.cpp
index 53a636e..8a8496f 100644
--- a/js/src/jsgc.cpp
+++ b/js/src/jsgc.cpp
@@ -10,7 +10,7 @@

 #include "prmjtime.h"

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/Util.h"

 /*
@@ -89,7 +89,7 @@ using namespace js;
 using namespace js::gc;

 using mozilla::ArrayEnd;
-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;
 using mozilla::Maybe;

 /* Perform a Full GC every 20 seconds if MaybeGC is called */
@@ -300,7 +300,7 @@ Arena::finalize(FreeOp *fop, AllocKind thingKind, size_t
    thingSize)
     FreeSpan *newListTail = &newListHead;
     uintptr_t newFreeSpanStart = 0;
     bool allClear = true;
-    DebugOnly<size_t> nmarked = 0;
+    DebugOnlyTor<size_t> nmarked = 0;
     for (;; thing += thingSize) {
         JS_ASSERT(thing <= lastByte + 1);
```

```
2075          if (thing == nextFree.first) {
2076 @@ -612,7 +612,7 @@ Chunk::prepareToBeFreed(JSRuntime *rt)
2077       rt->gcNumArenasFreeCommitted -= info.numArenasFreeCommitted;
2078       rt->gcStats.count(gcstats::STAT_DESTROY_CHUNK);
2079
2080 -#ifdef DEBUG
2081 +#ifndef TOR_NASSERT
2082       /*
2083        * Let FreeChunkList detect a missing prepareToBeFreed call before it
2084        * frees chunk.
2085 @@ -1774,7 +1774,7 @@ void
2086  GCMarker::checkZone(void *p)
2087  {
2088       JS_ASSERT(started);
2089 -     DebugOnly<Cell *> cell = static_cast<Cell *>(p);
2090 +     DebugOnlyTor<Cell *> cell = static_cast<Cell *>(p);
2091       JS_ASSERT_IF(cell->isTenured(), cell->tenuredZone()->isCollecting());
2092  }
2093  #endif
2094 diff --git a/js/src/jsgc.h b/js/src/jsgc.h
2095 index 4bf5c2f..92eb1a4 100644
2096 --- a/js/src/jsgc.h
2097 +++ b/js/src/jsgc.h
2098 @@ -9,7 +9,7 @@
2099  #ifndef jsgc_h
2100  #define jsgc_h
2101
2102 -#include "mozilla/DebugOnly.h"
2103 +#include "mozilla/DebugOnlyTor.h"
2104  #include "mozilla/Util.h"
2105
2106  #include "jsalloc.h"
2107 @@ -1138,12 +1138,12 @@ struct GCMarker : public JSTracer {
2108       /* The color is only applied to objects and functions. */
2109       uint32_t color;
2110
2111 -     mozilla::DebugOnly<bool> started;
2112 +     mozilla::DebugOnlyTor<bool> started;
2113
2114       /* Pointer to the top of the stack of arenas we are delaying marking on. */
2115       js::gc::ArenaHeader *unmarkedArenaStackTop;
2116       /* Count of arenas that are currently in the stack. */
2117 -     mozilla::DebugOnly<size_t> markLaterArenas;
2118 +     mozilla::DebugOnlyTor<size_t> markLaterArenas;
2119
2120       bool grayFailed;
2121  };
2122 diff --git a/js/src/jsgcinlines.h b/js/src/jsgcinlines.h
2123 index 7e95862..e2880ea 100644
2124 --- a/js/src/jsgcinlines.h
2125 +++ b/js/src/jsgcinlines.h
```

```
@@ -361,7 +361,7 @@ class CellIter : public CellIterImpl
 {
     ArenaLists *lists;
     AllocKind kind;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     size_t *counter;
 #endif
   public:
@@ -386,7 +386,7 @@ class CellIter : public CellIterImpl
             JS_ASSERT(!zone->rt->isHeapBusy());
             lists->copyFreeListToArena(kind);
         }
-#ifdef DEBUG
+#ifndef TOR_NASSERT
         counter = &zone->rt->noGCOrAllocationCheck;
         ++*counter;
 #endif
@@ -394,7 +394,7 @@ class CellIter : public CellIterImpl
     }

     ~CellIter() {
-#ifdef DEBUG
+#ifndef TOR_NASSERT
         JS_ASSERT(*counter > 0);
         --*counter;
 #endif
diff --git a/js/src/jsinfer.cpp b/js/src/jsinfer.cpp
index e961f11..bd4850b 100644
--- a/js/src/jsinfer.cpp
+++ b/js/src/jsinfer.cpp
@@ -6,7 +6,7 @@

 #include "jsinfer.h"

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/PodOperations.h"

 #include "jsapi.h"
@@ -47,7 +47,7 @@ using namespace js::gc;
 using namespace js::types;
 using namespace js::analyze;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;
 using mozilla::PodArrayZero;
 using mozilla::PodCopy;
 using mozilla::PodZero;
@@ -119,7 +119,7 @@ static bool InferSpewActive(SpewChannel channel)
     return active[channel];
```

```
 }

-#ifdef DEBUG
+#ifndef TOR_NASSERT

 static bool InferSpewColorable()
 {
@@ -1768,7 +1768,7 @@ StackTypeSet::getKnownTypeTag()
       * that the exact tag is unknown, as it will stay unknown as more types are
       * added to the set.
       */
-    DebugOnly<bool> empty = flags == 0 && baseObjectCount() == 0;
+    DebugOnlyTor<bool> empty = flags == 0 && baseObjectCount() == 0;
     JS_ASSERT_IF(empty, type == JSVAL_TYPE_UNKNOWN);

     return type;
@@ -1795,7 +1795,7 @@ HeapTypeSet::getKnownTypeTag(JSContext *cx)
       * that the exact tag is unknown, as it will stay unknown as more types are
       * added to the set.
       */
-    DebugOnly<bool> empty = flags == 0 && baseObjectCount() == 0;
+    DebugOnlyTor<bool> empty = flags == 0 && baseObjectCount() == 0;
     JS_ASSERT_IF(empty, type == JSVAL_TYPE_UNKNOWN);

     return type;
@@ -6003,7 +6003,7 @@ TypeObjectEntry::match(TypeObject *key, const Lookup &lookup)
     return key->proto == lookup.proto.raw() && key->clasp == lookup.clasp;
 }

-#ifdef DEBUG
+#ifndef TOR_NASSERT
 bool
 JSObject::hasNewType(Class *clasp, TypeObject *type)
 {
diff --git a/js/src/jsinfer.h b/js/src/jsinfer.h
index 61476d8..8f9f47d 100644
--- a/js/src/jsinfer.h
+++ b/js/src/jsinfer.h
@@ -1475,7 +1475,7 @@ enum SpewChannel {
     SPEW_COUNT
 };

-#ifdef DEBUG
+#ifndef TOR_NASSERT

 const char * InferSpewColorReset();
 const char * InferSpewColor(TypeConstraint *constraint);
diff --git a/js/src/jsinferinlines.h b/js/src/jsinferinlines.h
index d4c57a1..f3bcb86 100644
--- a/js/src/jsinferinlines.h
+++ b/js/src/jsinferinlines.h
```

```
@@ -122,7 +122,7 @@ CompilerOutput::isValid() const
     if (!script)
         return false;

-#if defined(DEBUG) && defined(JS_ION)
+#if !defined(TOR_NASSERT) && defined(JS_ION)
     TypeCompartment &types = script->compartment()->types;
 #endif

diff --git a/js/src/jsmemorymetrics.cpp b/js/src/jsmemorymetrics.cpp
index 5851e0c..7291799 100644
--- a/js/src/jsmemorymetrics.cpp
+++ b/js/src/jsmemorymetrics.cpp
@@ -6,7 +6,7 @@

 #include "js/MemoryMetrics.h"

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "jsapi.h"
 #include "jscntxt.h"
@@ -21,7 +21,7 @@

 #include "jsobjinlines.h"

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;

 using namespace js;

@@ -328,7 +328,7 @@ JS::CollectRuntimeStats(JSRuntime *rt, RuntimeStats *rtStats,
     ObjectPrivateVisit
     // Take the "explicit/js/runtime/" measurements.
     rt->sizeOfIncludingThis(rtStats->mallocSizeOf_, &rtStats->runtime);

-    DebugOnly<size_t> totalArenaSize = 0;
+    DebugOnlyTor<size_t> totalArenaSize = 0;

     rtStats->gcHeapGcThings = 0;
     for (size_t i = 0; i < rtStats->zoneStatsVector.length(); i++) {
@@ -336,7 +336,7 @@ JS::CollectRuntimeStats(JSRuntime *rt, RuntimeStats *rtStats,
     ObjectPrivateVisit

         rtStats->zTotals.add(zStats);
         rtStats->gcHeapGcThings += zStats.GCHeapThingsSize();
-#ifdef DEBUG
+#ifndef TOR_NASSERT
         totalArenaSize += zStats.gcHeapArenaAdmin + zStats.gcHeapUnusedGcThings;
 #endif
     }
```

```
@@ -348,7 +348,7 @@ JS::CollectRuntimeStats(JSRuntime *rt, RuntimeStats *rtStats,
    ObjectPrivateVisit
         rtStats->gcHeapGcThings += cStats.GCHeapThingsSize();
     }

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     totalArenaSize += rtStats->gcHeapGcThings;
     JS_ASSERT(totalArenaSize % gc::ArenaSize == 0);
 #endif
diff --git a/js/src/jsobj.h b/js/src/jsobj.h
index 7e4e534..ebfee18 100644
--- a/js/src/jsobj.h
+++ b/js/src/jsobj.h
@@ -417,7 +417,7 @@ class JSObject : public js::ObjectImpl

     js::types::TypeObject *getNewType(JSContext *cx, js::Class *clasp, JSFunction *
         fun = NULL);

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     bool hasNewType(js::Class *clasp, js::types::TypeObject *newType);
 #endif

diff --git a/js/src/jsonparser.h b/js/src/jsonparser.h
index ad4823d..8f1c691 100644
--- a/js/src/jsonparser.h
+++ b/js/src/jsonparser.h
@@ -100,7 +100,7 @@ class MOZ_STACK_CLASS JSONParser : private AutoGCRooter
     Vector<ElementVector*, 5> freeElements;
     Vector<PropertyVector*, 5> freeProperties;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     Token lastToken;
 #endif

@@ -120,7 +120,7 @@ class MOZ_STACK_CLASS JSONParser : private AutoGCRooter
         stack(cx),
         freeElements(cx),
         freeProperties(cx)
-#ifdef DEBUG
+#ifndef TOR_NASSERT
       , lastToken(Error)
 #endif
     {
@@ -162,7 +162,7 @@ class MOZ_STACK_CLASS JSONParser : private AutoGCRooter
     Token token(Token t) {
         JS_ASSERT(t != String);
         JS_ASSERT(t != Number);
-#ifdef DEBUG
```

```
+#ifndef TOR_NASSERT
          lastToken = t;
 #endif
          return t;
@@ -170,7 +170,7 @@ class MOZ_STACK_CLASS JSONParser : private AutoGCRooter

      Token stringToken(JSString *str) {
          this->v = StringValue(str);
-#ifdef DEBUG
+#ifndef TOR_NASSERT
          lastToken = String;
 #endif
          return String;
@@ -178,7 +178,7 @@ class MOZ_STACK_CLASS JSONParser : private AutoGCRooter

      Token numberToken(double d) {
          this->v = NumberValue(d);
-#ifdef DEBUG
+#ifndef TOR_NASSERT
          lastToken = Number;
 #endif
          return Number;
diff --git a/js/src/jsopcode.cpp b/js/src/jsopcode.cpp
index facb4cf..313735a 100644
--- a/js/src/jsopcode.cpp
+++ b/js/src/jsopcode.cpp
@@ -735,7 +735,7 @@ Sprinter::realloc_(size_t newSize)

 Sprinter::Sprinter(JSContext *cx)
    : context(cx),
-#ifdef DEBUG
+#ifndef TOR_NASSERT
      initialized(false),
 #endif
      base(NULL), size(0), offset(0), reportedOOM(false)
@@ -743,7 +743,7 @@ Sprinter::Sprinter(JSContext *cx)

 Sprinter::~Sprinter()
 {
-#ifdef DEBUG
+#ifndef TOR_NASSERT
      if (initialized)
          checkInvariants();
 #endif
@@ -757,7 +757,7 @@ Sprinter::init()
      base = (char *) context->malloc_(DefaultSize);
      if (!base)
          return false;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
      initialized = true;
```

```
 #endif
     *base = 0;
diff --git a/js/src/jsopcode.h b/js/src/jsopcode.h
index 77f5141..aa4be3b 100644
--- a/js/src/jsopcode.h
+++ b/js/src/jsopcode.h
@@ -316,7 +316,7 @@ class Sprinter

   private:
     static const size_t     DefaultSize;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     bool                    initialized;    /* true if this is initialized, use for
         debug builds */
 #endif
     char                    *base;          /* malloc'd buffer address */
@@ -529,7 +529,7 @@ class PCCounts
 {
     friend class ::JSScript;
     double *counts;
-#ifdef DEBUG
+#ifndef TOR_NASSERT
     size_t capacity;
 #elif JS_BITS_PER_WORD == 32
     void *padding;
diff --git a/js/src/jsreflect.cpp b/js/src/jsreflect.cpp
index 59f6b89..cb29ecb 100644
--- a/js/src/jsreflect.cpp
+++ b/js/src/jsreflect.cpp
@@ -10,7 +10,7 @@

 #include <stdlib.h>

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/Util.h"

 #include "jspubtd.h"
@@ -30,7 +30,7 @@ using namespace js;
 using namespace js::frontend;

 using mozilla::ArrayLength;
-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;

 char const * const js::aopNames[] = {
     "=",    /* AOP_ASSIGN */
@@ -1480,7 +1480,7 @@ class ASTSerializer
     JSContext           *cx;
     Parser<FullParseHandler> *parser;
     NodeBuilder         builder;
```

```
-    DebugOnly<uint32_t> lineno;
+    DebugOnlyTor<uint32_t> lineno;

     Value unrootedAtomContents(JSAtom *atom) {
         return StringValue(atom ? atom : cx->names().empty);
diff --git a/js/src/jsscript.h b/js/src/jsscript.h
index 9b4c5c1..8d00773 100644
--- a/js/src/jsscript.h
+++ b/js/src/jsscript.h
@@ -470,7 +470,7 @@ class JSScript : public js::gc::Cell
                                    * or has had backedges taken. Reset if the
                                    * script's JIT code is forcibly discarded. */

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     // Unique identifier within the compartment for this script, used for
     // printing analysis information.
     uint32_t        id_;
@@ -762,7 +762,7 @@ class JSScript : public js::gc::Cell
     /* Return whether this script was compiled for 'eval' */
     bool isForEval() { return isCachedEval || isActiveEval; }

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     unsigned id();
 #else
     unsigned id() { return 0; }
diff --git a/js/src/jstypedarray.cpp b/js/src/jstypedarray.cpp
index 9d02d06..b85e768 100644
--- a/js/src/jstypedarray.cpp
+++ b/js/src/jstypedarray.cpp
@@ -738,7 +738,7 @@ ArrayBufferObject::obj_trace(JSTracer *trc, JSObject *obj)
                 SetBufferLink(firstView, *bufList);
                 *bufList = obj;
             } else {
-#ifdef DEBUG
+#ifndef TOR_NASSERT
                 bool found = false;
                 for (JSObject *p = obj->compartment()->gcLiveArrayBuffers; p; p =
                     BufferLink(p)) {
                     if (p == obj)
@@ -1808,7 +1808,7 @@ class TypedArrayTemplate
             return NULL;
         obj->setLastPropertyInfallible(empty);

-#ifdef DEBUG
+#ifndef TOR_NASSERT
         uint32_t bufferByteLength = buffer->byteLength();
         uint32_t arrayByteLength = static_cast<uint32_t>(byteLengthValue(obj).
             toInt32());
```

```
2475          uint32_t arrayByteOffset = static_cast<uint32_t>(byteOffsetValue(obj).
                  toInt32());
2476 @@ -2045,7 +2045,7 @@ class TypedArrayTemplate
2477          uint32_t byteSrc = srcBegin * sizeof(NativeType);
2478          uint32_t byteSize = nelts * sizeof(NativeType);
2479
2480 -#ifdef DEBUG
2481 +#ifndef TOR_NASSERT
2482          uint32_t viewByteLength = byteLengthValue(tarray).toInt32();
2483          JS_ASSERT(byteDest <= viewByteLength);
2484          JS_ASSERT(byteSrc <= viewByteLength);
2485 @@ -2369,7 +2369,7 @@ class TypedArrayTemplate
2486          SkipRoot skipDest(cx, &dest);
2487          SkipRoot skipSrc(cx, &src);
2488
2489 -#ifdef DEBUG
2490 +#ifndef TOR_NASSERT
2491          JSRuntime *runtime = cx->runtime();
2492          uint64_t gcNumber = runtime->gcNumber;
2493  #endif
2494 diff --git a/js/src/jsutil.cpp b/js/src/jsutil.cpp
2495 index bcab124..e29d3fb 100644
2496 --- a/js/src/jsutil.cpp
2497 +++ b/js/src/jsutil.cpp
2498 @@ -8,7 +8,7 @@
2499
2500  #include "jsutil.h"
2501
2502 -#include "mozilla/Assertions.h"
2503 +#include "mozilla/AssertionsTor.h"
2504  #include "mozilla/PodOperations.h"
2505
2506  #include <stdio.h>
2507 @@ -154,8 +154,8 @@ JS_STATIC_ASSERT(sizeof(void *) == sizeof(void (*)()));
2508  JS_PUBLIC_API(void)
2509  JS_Assert(const char *s, const char *file, int ln)
2510  {
2511 -    MOZ_ReportAssertionFailure(s, file, ln);
2512 -    MOZ_CRASH();
2513 +    TBB_MOZ_ReportAssertionFailure(s, file, ln);
2514 +    TBB_MOZ_CRASH();
2515  }
2516
2517  #ifdef JS_BASIC_STATS
2518 diff --git a/js/src/jsworkers.cpp b/js/src/jsworkers.cpp
2519 index 57b16ea..277534b 100644
2520 --- a/js/src/jsworkers.cpp
2521 +++ b/js/src/jsworkers.cpp
2522 @@ -6,7 +6,7 @@
2523
2524  #include "jsworkers.h"
```

```
2525
2526 -#include "mozilla/DebugOnly.h"
2527 +#include "mozilla/DebugOnlyTor.h"
2528
2529  #include "prmjtime.h"
2530
2531 @@ -18,7 +18,7 @@
2532
2533  using namespace js;
2534
2535 -using mozilla::DebugOnly;
2536 +using mozilla::DebugOnlyTor;
2537
2538  #ifdef JS_PARALLEL_COMPILATION
2539
2540 @@ -230,7 +230,7 @@ WorkerThreadState::lock()
2541  {
2542      JS_ASSERT(!isLocked());
2543      PR_Lock(workerLock);
2544 -#ifdef DEBUG
2545 +#ifndef TOR_NASSERT
2546      lockOwner = PR_GetCurrentThread();
2547  #endif
2548  }
2549 @@ -239,13 +239,13 @@ void
2550  WorkerThreadState::unlock()
2551  {
2552      JS_ASSERT(isLocked());
2553 -#ifdef DEBUG
2554 +#ifndef TOR_NASSERT
2555      lockOwner = NULL;
2556  #endif
2557      PR_Unlock(workerLock);
2558  }
2559
2560 -#ifdef DEBUG
2561 +#ifndef TOR_NASSERT
2562  bool
2563  WorkerThreadState::isLocked()
2564  {
2565 @@ -257,14 +257,14 @@ void
2566  WorkerThreadState::wait(CondVar which, uint32_t millis)
2567  {
2568      JS_ASSERT(isLocked());
2569 -#ifdef DEBUG
2570 +#ifndef TOR_NASSERT
2571      lockOwner = NULL;
2572  #endif
2573 -    DebugOnly<PRStatus> status =
2574 +    DebugOnlyTor<PRStatus> status =
2575          PR_WaitCondVar((which == MAIN) ? mainWakeup : helperWakeup,
```

```
2576                         millis ? PR_MillisecondsToInterval(millis) :
                                 PR_INTERVAL_NO_TIMEOUT);
2577      JS_ASSERT(status == PR_SUCCESS);
2578 -#ifdef DEBUG
2579 +#ifndef TOR_NASSERT
2580      lockOwner = PR_GetCurrentThread();
2581  #endif
2582  }
2583 @@ -389,7 +389,7 @@ WorkerThread::handleIonWorkload(WorkerThreadState &state)
2584
2585      ionBuilder = state.ionWorklist.popCopy();
2586
2587 -    DebugOnly<jit::ExecutionMode> executionMode = ionBuilder->info().executionMode()
     ;
2588 +    DebugOnlyTor<jit::ExecutionMode> executionMode = ionBuilder->info().
     executionMode();
2589      JS_ASSERT(GetIonScript(ionBuilder->script(), executionMode) ==
              ION_COMPILING_SCRIPT);
2590
2591      state.unlock();
2592 diff --git a/js/src/jsworkers.h b/js/src/jsworkers.h
2593 index f29aa81..c4ae0b9 100644
2594 --- a/js/src/jsworkers.h
2595 +++ b/js/src/jsworkers.h
2596 @@ -69,7 +69,7 @@ class WorkerThreadState
2597      void lock();
2598      void unlock();
2599
2600 -# ifdef DEBUG
2601 +#ifndef TOR_NASSERT
2602      bool isLocked();
2603  # endif
2604
2605 @@ -112,7 +112,7 @@ class WorkerThreadState
2606       */
2607      PRLock *workerLock;
2608
2609 -# ifdef DEBUG
2610 +#ifndef TOR_NASSERT
2611      PRThread *lockOwner;
2612  # endif
2613
2614 diff --git a/js/src/shell/js.cpp b/js/src/shell/js.cpp
2615 index 7aa9380..120b328 100644
2616 --- a/js/src/shell/js.cpp
2617 +++ b/js/src/shell/js.cpp
2618 @@ -13,7 +13,7 @@
2619  #include <stdlib.h>
2620  #include <string.h>
2621
2622 -#include "mozilla/DebugOnly.h"
```

```
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/GuardObjects.h"
 #include "mozilla/Util.h"

@@ -2899,7 +2899,7 @@ WatchdogMain(void *arg)
              uint64_t sleepDuration = PR_INTERVAL_NO_TIMEOUT;
              if (gWatchdogHasTimeout)
                  sleepDuration = PR_TicksPerSecond() / 10;
-             mozilla::DebugOnly<PRStatus> status =
+             mozilla::DebugOnlyTor<PRStatus> status =
                PR_WaitCondVar(gWatchdogWakeup, sleepDuration);
              JS_ASSERT(status == PR_SUCCESS);
          }
@@ -4537,7 +4537,7 @@ dom_genericSetter(JSContext* cx, unsigned argc, JS::Value *vp);
 static JSBool
 dom_genericMethod(JSContext *cx, unsigned argc, JS::Value *vp);

-#ifdef DEBUG
+#ifndef TOR_NASSERT
 static JSClass *GetDomClass();
 #endif

@@ -4628,7 +4628,7 @@ static JSClass dom_class = {
     JSCLASS_NO_INTERNAL_MEMBERS
 };

-#ifdef DEBUG
+#ifndef TOR_NASSERT
 static JSClass *GetDomClass() {
     return &dom_class;
 }
diff --git a/js/src/vm/GlobalObject.h b/js/src/vm/GlobalObject.h
index 1869ab9..2927367 100644
--- a/js/src/vm/GlobalObject.h
+++ b/js/src/vm/GlobalObject.h
@@ -7,7 +7,7 @@
 #ifndef vm_GlobalObject_h
 #define vm_GlobalObject_h

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "jsarray.h"
 #include "jsbool.h"
@@ -382,7 +382,7 @@ class GlobalObject : public JSObject
              return true;
          if (!cx->runtime()->cloneSelfHostedValue(cx, name, value))
              return false;
-         mozilla::DebugOnly<bool> ok = JS_DefinePropertyById(cx, holder, id, value,
     NULL, NULL, 0);
```

```
+        mozilla::DebugOnlyTor<bool> ok = JS_DefinePropertyById(cx, holder, id, value
    , NULL, NULL, 0);
         JS_ASSERT(ok);
         return true;
     }
diff --git a/js/src/vm/Interpreter.cpp b/js/src/vm/Interpreter.cpp
index 30a7627..a6af6ca 100644
--- a/js/src/vm/Interpreter.cpp
+++ b/js/src/vm/Interpreter.cpp
@@ -10,7 +10,7 @@

 #include "Interpreter.h"

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/FloatingPoint.h"
 #include "mozilla/PodOperations.h"

@@ -58,7 +58,7 @@ using namespace js;
 using namespace js::gc;
 using namespace js::types;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;
 using mozilla::PodCopy;

 /* Some objects (e.g., With) delegate 'this' to another object. */
@@ -1198,7 +1198,7 @@ Interpret(JSContext *cx, RunState &state)
     RootedId rootId0(cx);
     RootedShape rootShape0(cx);
     RootedScript rootScript0(cx);
-    DebugOnly<uint32_t> blockDepth;
+    DebugOnlyTor<uint32_t> blockDepth;

 #if JS_HAS_GENERATORS
     if (JS_UNLIKELY(regs.fp()->isGeneratorFrame())) {
diff --git a/js/src/vm/Monitor.h b/js/src/vm/Monitor.h
index 9aaa504..c814aa2 100644
--- a/js/src/vm/Monitor.h
+++ b/js/src/vm/Monitor.h
@@ -69,7 +69,7 @@ class AutoLockMonitor

     void wait() {
 #ifdef JS_THREADSAFE
-        mozilla::DebugOnly<PRStatus> status =
+        mozilla::DebugOnlyTor<PRStatus> status =
           PR_WaitCondVar(monitor.condVar_, PR_INTERVAL_NO_TIMEOUT);
         JS_ASSERT(status == PR_SUCCESS);
 #endif
diff --git a/js/src/vm/NumericConversions.h b/js/src/vm/NumericConversions.h
index 61511a0..a75dcbb 100644
```

```
--- a/js/src/vm/NumericConversions.h
+++ b/js/src/vm/NumericConversions.h
@@ -7,7 +7,7 @@
 #ifndef vm_NumericConversions_h
 #define vm_NumericConversions_h

-#include "mozilla/Assertions.h"
+#include "mozilla/AssertionsTor.h"
 #include "mozilla/Casting.h"
 #include "mozilla/FloatingPoint.h"
 #include "mozilla/TypeTraits.h"
@@ -38,7 +38,7 @@ template<typename ResultType>
 inline ResultType
 ToUintWidth(double d)
 {
-    MOZ_STATIC_ASSERT(mozilla::IsUnsigned<ResultType>::value,
+    TBB_MOZ_STATIC_ASSERT(mozilla::IsUnsigned<ResultType>::value,
                       "ResultType must be an unsigned type");

     uint64_t bits = mozilla::BitwiseCast<uint64_t>(d);
@@ -69,7 +69,7 @@ ToUintWidth(double d)
     // The significand contains the bits that will determine the final result.
     // Shift those bits left or right, according to the exponent, to their
     // locations in the unsigned binary representation of floor(abs(d)).
-    MOZ_STATIC_ASSERT(sizeof(ResultType) <= sizeof(uint64_t),
+    TBB_MOZ_STATIC_ASSERT(sizeof(ResultType) <= sizeof(uint64_t),
                       "Left-shifting below would lose upper bits");
     ResultType result = (exponent > mozilla::DoubleExponentShift)
                         ? ResultType(bits << (exponent - mozilla::
                             DoubleExponentShift))
@@ -113,7 +113,7 @@ template<typename ResultType>
 inline ResultType
 ToIntWidth(double d)
 {
-    MOZ_STATIC_ASSERT(mozilla::IsSigned<ResultType>::value,
+    TBB_MOZ_STATIC_ASSERT(mozilla::IsSigned<ResultType>::value,
                       "ResultType must be a signed type");

     const ResultType MaxValue = (1ULL << (CHAR_BIT * sizeof(ResultType) - 1)) - 1;
diff --git a/js/src/vm/ObjectImpl-inl.h b/js/src/vm/ObjectImpl-inl.h
index c5a4b4a..560be3d 100644
--- a/js/src/vm/ObjectImpl-inl.h
+++ b/js/src/vm/ObjectImpl-inl.h
@@ -7,7 +7,7 @@
 #ifndef vm_ObjectImpl_inl_h
 #define vm_ObjectImpl_inl_h

-#include "mozilla/Assertions.h"
+#include "mozilla/AssertionsTor.h"

 #include "jscompartment.h"
```

```
 #include "jsgc.h"
@@ -126,35 +126,35 @@ js::ObjectImpl::isExtensible() const
 inline uint32_t
 js::ObjectImpl::getDenseInitializedLength()
 {
-    MOZ_ASSERT(isNative());
+    TBB_MOZ_ASSERT(isNative());
     return getElementsHeader()->initializedLength;
 }

 inline uint32_t
 js::ObjectImpl::getDenseCapacity()
 {
-    MOZ_ASSERT(isNative());
+    TBB_MOZ_ASSERT(isNative());
     return getElementsHeader()->capacity;
 }

 inline js::HeapSlotArray
 js::ObjectImpl::getDenseElements()
 {
-    MOZ_ASSERT(isNative());
+    TBB_MOZ_ASSERT(isNative());
     return HeapSlotArray(elements);
 }

 inline const js::Value &
 js::ObjectImpl::getDenseElement(uint32_t idx)
 {
-    MOZ_ASSERT(isNative() && idx < getDenseInitializedLength());
+    TBB_MOZ_ASSERT(isNative() && idx < getDenseInitializedLength());
     return elements[idx];
 }

 inline bool
 js::ObjectImpl::containsDenseElement(uint32_t idx)
 {
-    MOZ_ASSERT(isNative());
+    TBB_MOZ_ASSERT(isNative());
     return idx < getDenseInitializedLength() && !elements[idx].isMagic(
         JS_ELEMENTS_HOLE);
 }

@@ -163,7 +163,7 @@ js::ObjectImpl::getSlotRangeUnchecked(uint32_t start, uint32_t
    length,
                                       HeapSlot **fixedStart, HeapSlot **fixedEnd,
                                       HeapSlot **slotsStart, HeapSlot **slotsEnd)
 {
-    MOZ_ASSERT(start + length >= start);
+    TBB_MOZ_ASSERT(start + length >= start);

```

```
2821      uint32_t fixed = numFixedSlots();
2822      if (start < fixed) {
2823  @@ -190,7 +190,7 @@ js::ObjectImpl::getSlotRange(uint32_t start, uint32_t length,
2824                               HeapSlot **fixedStart, HeapSlot **fixedEnd,
2825                               HeapSlot **slotsStart, HeapSlot **slotsEnd)
2826   {
2827  -    MOZ_ASSERT(slotInRange(start + length, SENTINEL_ALLOWED));
2828  +    TBB_MOZ_ASSERT(slotInRange(start + length, SENTINEL_ALLOWED));
2829      getSlotRangeUnchecked(start, length, fixedStart, fixedEnd, slotsStart, slotsEnd)
           ;
2830   }
2831
2832  @@ -220,20 +220,20 @@ js::ObjectImpl::isProxy() const
2833   inline js::HeapSlot &
2834   js::ObjectImpl::nativeGetSlotRef(uint32_t slot)
2835   {
2836  -    MOZ_ASSERT(isNative());
2837  -    MOZ_ASSERT(slot < slotSpan());
2838  +    TBB_MOZ_ASSERT(isNative());
2839  +    TBB_MOZ_ASSERT(slot < slotSpan());
2840      return getSlotRef(slot);
2841   }
2842
2843   inline const js::Value &
2844   js::ObjectImpl::nativeGetSlot(uint32_t slot) const
2845   {
2846  -    MOZ_ASSERT(isNative());
2847  -    MOZ_ASSERT(slot < slotSpan());
2848  +    TBB_MOZ_ASSERT(isNative());
2849  +    TBB_MOZ_ASSERT(slot < slotSpan());
2850      return getSlot(slot);
2851   }
2852
2853  -#ifdef DEBUG
2854  +#ifndef TOR_NASSERT
2855   inline bool
2856   IsObjectValueInCompartment(js::Value v, JSCompartment *comp)
2857   {
2858  @@ -246,32 +246,32 @@ IsObjectValueInCompartment(js::Value v, JSCompartment *comp)
2859   inline void
2860   js::ObjectImpl::setSlot(uint32_t slot, const js::Value &value)
2861   {
2862  -    MOZ_ASSERT(slotInRange(slot));
2863  -    MOZ_ASSERT(IsObjectValueInCompartment(value, asObjectPtr()->compartment()));
2864  +    TBB_MOZ_ASSERT(slotInRange(slot));
2865  +    TBB_MOZ_ASSERT(IsObjectValueInCompartment(value, asObjectPtr()->compartment()));
2866      getSlotRef(slot).set(this->asObjectPtr(), HeapSlot::Slot, slot, value);
2867   }
2868
2869   inline void
2870   js::ObjectImpl::setCrossCompartmentSlot(uint32_t slot, const js::Value &value)
```

```
 {
-    MOZ_ASSERT(slotInRange(slot));
+    TBB_MOZ_ASSERT(slotInRange(slot));
     getSlotRef(slot).set(this->asObjectPtr(), HeapSlot::Slot, slot, value);
 }

 inline void
 js::ObjectImpl::initSlot(uint32_t slot, const js::Value &value)
 {
-    MOZ_ASSERT(getSlot(slot).isUndefined());
-    MOZ_ASSERT(slotInRange(slot));
-    MOZ_ASSERT(IsObjectValueInCompartment(value, asObjectPtr()->compartment()));
+    TBB_MOZ_ASSERT(getSlot(slot).isUndefined());
+    TBB_MOZ_ASSERT(slotInRange(slot));
+    TBB_MOZ_ASSERT(IsObjectValueInCompartment(value, asObjectPtr()->compartment()));
     initSlotUnchecked(slot, value);
 }

 inline void
 js::ObjectImpl::initCrossCompartmentSlot(uint32_t slot, const js::Value &value)
 {
-    MOZ_ASSERT(getSlot(slot).isUndefined());
-    MOZ_ASSERT(slotInRange(slot));
+    TBB_MOZ_ASSERT(getSlot(slot).isUndefined());
+    TBB_MOZ_ASSERT(slotInRange(slot));
     initSlotUnchecked(slot, value);
 }

@@ -284,14 +284,14 @@ js::ObjectImpl::initSlotUnchecked(uint32_t slot, const js::
    Value &value)
 inline void
 js::ObjectImpl::setFixedSlot(uint32_t slot, const js::Value &value)
 {
-    MOZ_ASSERT(slot < numFixedSlots());
+    TBB_MOZ_ASSERT(slot < numFixedSlots());
     fixedSlots()[slot].set(this->asObjectPtr(), HeapSlot::Slot, slot, value);
 }

 inline void
 js::ObjectImpl::initFixedSlot(uint32_t slot, const js::Value &value)
 {
-    MOZ_ASSERT(slot < numFixedSlots());
+    TBB_MOZ_ASSERT(slot < numFixedSlots());
     fixedSlots()[slot].init(this->asObjectPtr(), HeapSlot::Slot, slot, value);
 }

@@ -343,7 +343,7 @@ js::ObjectImpl::dynamicSlotsCount(uint32_t nfixed, uint32_t span)
         return SLOT_CAPACITY_MIN;

     uint32_t slots = RoundUpPow2(span);
-    MOZ_ASSERT(slots >= span);
```

```
+    TBB_MOZ_ASSERT(slots >= span);
     return slots;
 }

@@ -366,10 +366,10 @@ js::ObjectImpl::readBarrier(ObjectImpl *obj)
 #ifdef JSGC_INCREMENTAL
     Zone *zone = obj->zone();
     if (zone->needsBarrier()) {
-        MOZ_ASSERT(!zone->rt->isHeapMajorCollecting());
+        TBB_MOZ_ASSERT(!zone->rt->isHeapMajorCollecting());
         JSObject *tmp = obj->asObjectPtr();
         MarkObjectUnbarriered(zone->barrierTracer(), &tmp, "read barrier");
-        MOZ_ASSERT(tmp == obj->asObjectPtr());
+        TBB_MOZ_ASSERT(tmp == obj->asObjectPtr());
     }
 #endif
 }
@@ -407,10 +407,10 @@ js::ObjectImpl::writeBarrierPre(ObjectImpl *obj)

     Zone *zone = obj->zone();
     if (zone->needsBarrier()) {
-        MOZ_ASSERT(!zone->rt->isHeapMajorCollecting());
+        TBB_MOZ_ASSERT(!zone->rt->isHeapMajorCollecting());
         JSObject *tmp = obj->asObjectPtr();
         MarkObjectUnbarriered(zone->barrierTracer(), &tmp, "write barrier");
-        MOZ_ASSERT(tmp == obj->asObjectPtr());
+        TBB_MOZ_ASSERT(tmp == obj->asObjectPtr());
     }
 #endif
 }
diff --git a/js/src/vm/ObjectImpl.cpp b/js/src/vm/ObjectImpl.cpp
index b1ce275..c366708 100644
--- a/js/src/vm/ObjectImpl.cpp
+++ b/js/src/vm/ObjectImpl.cpp
@@ -284,7 +284,7 @@ js::ObjectImpl::copySlotRange(uint32_t start, const Value *vector
    , uint32_t leng
         sp->set(zone, this->asObjectPtr(), HeapSlot::Slot, start++, *vector++);
 }

-#ifdef DEBUG
+#ifndef TOR_NASSERT
 bool
 js::ObjectImpl::slotInRange(uint32_t slot, SentinelAllowed sentinel) const
 {
@@ -293,7 +293,7 @@ js::ObjectImpl::slotInRange(uint32_t slot, SentinelAllowed
    sentinel) const
         return slot <= capacity;
     return slot < capacity;
 }
-#endif /* DEBUG */
+#endif /* TOR_NASSERT */
```

```
2970
2971  // See bug 844580.
2972  #if defined(_MSC_VER)
2973 diff --git a/js/src/vm/ObjectImpl.h b/js/src/vm/ObjectImpl.h
2974 index 8eba5da..4edb6bb 100644
2975 --- a/js/src/vm/ObjectImpl.h
2976 +++ b/js/src/vm/ObjectImpl.h
2977 @@ -7,7 +7,7 @@
2978  #ifndef vm_ObjectImpl_h
2979  #define vm_ObjectImpl_h
2980
2981 -#include "mozilla/Assertions.h"
2982 +#include "mozilla/AssertionsTor.h"
2983  #include "mozilla/GuardObjects.h"
2984  #include "mozilla/StandardInteger.h"
2985
2986 @@ -55,11 +55,11 @@ class PropertyId
2987
2988    public:
2989      bool isName() const {
2990 -        MOZ_ASSERT(JSID_IS_STRING(id) || JSID_IS_SPECIAL(id));
2991 +        TBB_MOZ_ASSERT(JSID_IS_STRING(id) || JSID_IS_SPECIAL(id));
2992          return JSID_IS_STRING(id);
2993      }
2994      bool isSpecial() const {
2995 -        MOZ_ASSERT(JSID_IS_STRING(id) || JSID_IS_SPECIAL(id));
2996 +        TBB_MOZ_ASSERT(JSID_IS_STRING(id) || JSID_IS_SPECIAL(id));
2997          return !isName();
2998      }
2999
3000 @@ -195,17 +195,17 @@ struct PropDesc {
3001
3002      bool isUndefined() const { return isUndefined_; }
3003
3004 -    bool hasGet() const { MOZ_ASSERT(!isUndefined()); return hasGet_; }
3005 -    bool hasSet() const { MOZ_ASSERT(!isUndefined()); return hasSet_; }
3006 -    bool hasValue() const { MOZ_ASSERT(!isUndefined()); return hasValue_; }
3007 -    bool hasWritable() const { MOZ_ASSERT(!isUndefined()); return hasWritable_; }
3008 -    bool hasEnumerable() const { MOZ_ASSERT(!isUndefined()); return hasEnumerable_;
     }
3009 -    bool hasConfigurable() const { MOZ_ASSERT(!isUndefined()); return
     hasConfigurable_; }
3010 +    bool hasGet() const { TBB_MOZ_ASSERT(!isUndefined()); return hasGet_; }
3011 +    bool hasSet() const { TBB_MOZ_ASSERT(!isUndefined()); return hasSet_; }
3012 +    bool hasValue() const { TBB_MOZ_ASSERT(!isUndefined()); return hasValue_; }
3013 +    bool hasWritable() const { TBB_MOZ_ASSERT(!isUndefined()); return hasWritable_;
     }
3014 +    bool hasEnumerable() const { TBB_MOZ_ASSERT(!isUndefined()); return
     hasEnumerable_; }
3015 +    bool hasConfigurable() const { TBB_MOZ_ASSERT(!isUndefined()); return
     hasConfigurable_; }
```

```
3016
3017 -    Value pd() const { MOZ_ASSERT(!isUndefined()); return pd_; }
3018 +    Value pd() const { TBB_MOZ_ASSERT(!isUndefined()); return pd_; }
3019      void clearPd() { pd_ = UndefinedValue(); }
3020
3021 -    uint8_t attributes() const { MOZ_ASSERT(!isUndefined()); return attrs; }
3022 +    uint8_t attributes() const { TBB_MOZ_ASSERT(!isUndefined()); return attrs; }
3023
3024      /* 8.10.1 IsAccessorDescriptor(desc) */
3025      bool isAccessorDescriptor() const {
3026 @@ -223,47 +223,47 @@ struct PropDesc {
3027      }
3028
3029      bool configurable() const {
3030 -        MOZ_ASSERT(!isUndefined());
3031 -        MOZ_ASSERT(hasConfigurable());
3032 +        TBB_MOZ_ASSERT(!isUndefined());
3033 +        TBB_MOZ_ASSERT(hasConfigurable());
3034          return (attrs & JSPROP_PERMANENT) == 0;
3035      }
3036
3037      bool enumerable() const {
3038 -        MOZ_ASSERT(!isUndefined());
3039 -        MOZ_ASSERT(hasEnumerable());
3040 +        TBB_MOZ_ASSERT(!isUndefined());
3041 +        TBB_MOZ_ASSERT(hasEnumerable());
3042          return (attrs & JSPROP_ENUMERATE) != 0;
3043      }
3044
3045      bool writable() const {
3046 -        MOZ_ASSERT(!isUndefined());
3047 -        MOZ_ASSERT(hasWritable());
3048 +        TBB_MOZ_ASSERT(!isUndefined());
3049 +        TBB_MOZ_ASSERT(hasWritable());
3050          return (attrs & JSPROP_READONLY) == 0;
3051      }
3052
3053      HandleValue value() const {
3054 -        MOZ_ASSERT(hasValue());
3055 +        TBB_MOZ_ASSERT(hasValue());
3056          return HandleValue::fromMarkedLocation(&value_);
3057      }
3058
3059      JSObject * getterObject() const {
3060 -        MOZ_ASSERT(!isUndefined());
3061 -        MOZ_ASSERT(hasGet());
3062 +        TBB_MOZ_ASSERT(!isUndefined());
3063 +        TBB_MOZ_ASSERT(hasGet());
3064          return get_.isUndefined() ? NULL : &get_.toObject();
3065      }
3066      JSObject * setterObject() const {
```

```
-         MOZ_ASSERT(!isUndefined());
-         MOZ_ASSERT(hasSet());
+         TBB_MOZ_ASSERT(!isUndefined());
+         TBB_MOZ_ASSERT(hasSet());
          return set_.isUndefined() ? NULL : &set_.toObject();
      }

      HandleValue getterValue() const {
-         MOZ_ASSERT(!isUndefined());
-         MOZ_ASSERT(hasGet());
+         TBB_MOZ_ASSERT(!isUndefined());
+         TBB_MOZ_ASSERT(hasGet());
          return HandleValue::fromMarkedLocation(&get_);
      }
      HandleValue setterValue() const {
-         MOZ_ASSERT(!isUndefined());
-         MOZ_ASSERT(hasSet());
+         TBB_MOZ_ASSERT(!isUndefined());
+         TBB_MOZ_ASSERT(hasSet());
          return HandleValue::fromMarkedLocation(&set_);
      }

@@ -407,13 +407,13 @@ class ElementsHeader
      };

      void staticAsserts() {
-         MOZ_STATIC_ASSERT(sizeof(ElementsHeader) == ValuesPerHeader * sizeof(Value),
+         TBB_MOZ_STATIC_ASSERT(sizeof(ElementsHeader) == ValuesPerHeader * sizeof(
    Value),
                            "Elements size and values-per-Elements mismatch");
      }

    public:
      ElementsKind kind() const {
-         MOZ_ASSERT(type <= ArrayBufferElements);
+         TBB_MOZ_ASSERT(type <= ArrayBufferElements);
          return ElementsKind(type);
      }

@@ -454,17 +454,17 @@ class DenseElementsHeader : public ElementsHeader
 {
    public:
      uint32_t capacity() const {
-         MOZ_ASSERT(ElementsHeader::isDenseElements());
+         TBB_MOZ_ASSERT(ElementsHeader::isDenseElements());
          return dense.capacity;
      }

      uint32_t initializedLength() const {
-         MOZ_ASSERT(ElementsHeader::isDenseElements());
+         TBB_MOZ_ASSERT(ElementsHeader::isDenseElements());
```

```
3117          return dense.initializedLength;
3118      }
3119
3120      uint32_t length() const {
3121  -       MOZ_ASSERT(ElementsHeader::isDenseElements());
3122  +       TBB_MOZ_ASSERT(ElementsHeader::isDenseElements());
3123          return ElementsHeader::length;
3124      }
3125
3126  @@ -490,12 +490,12 @@ class SparseElementsHeader : public ElementsHeader
3127   {
3128     public:
3129      Shape *shape() {
3130  -       MOZ_ASSERT(ElementsHeader::isSparseElements());
3131  +       TBB_MOZ_ASSERT(ElementsHeader::isSparseElements());
3132          return sparse.shape;
3133      }
3134
3135      uint32_t length() const {
3136  -       MOZ_ASSERT(ElementsHeader::isSparseElements());
3137  +       TBB_MOZ_ASSERT(ElementsHeader::isSparseElements());
3138          return ElementsHeader::length;
3139      }
3140
3141  @@ -588,7 +588,7 @@ struct uint8_clamped {
3142      }
3143
3144      void staticAsserts() {
3145  -       MOZ_STATIC_ASSERT(sizeof(uint8_clamped) == 1,
3146  +       TBB_MOZ_STATIC_ASSERT(sizeof(uint8_clamped) == 1,
3147                            "uint8_clamped must be layout-compatible with uint8_t");
3148      }
3149   };
3150  @@ -607,21 +607,21 @@ template <typename T>
3151   class TypedElementsHeader : public ElementsHeader
3152   {
3153      T getElement(uint32_t index) {
3154  -       MOZ_ASSERT(index < length());
3155  +       TBB_MOZ_ASSERT(index < length());
3156          return reinterpret_cast<T *>(this + 1)[index];
3157      }
3158
3159      inline void assign(uint32_t index, double d);
3160
3161      void setElement(uint32_t index, T value) {
3162  -       MOZ_ASSERT(index < length());
3163  +       TBB_MOZ_ASSERT(index < length());
3164          reinterpret_cast<T *>(this + 1)[index] = value;
3165      }
3166
3167     public:
```

```
3168      uint32_t length() const {
3169  -         MOZ_ASSERT(Uint8Elements <= kind());
3170  -         MOZ_ASSERT(kind() <= Float64Elements);
3171  +         TBB_MOZ_ASSERT(Uint8Elements <= kind());
3172  +         TBB_MOZ_ASSERT(kind() <= Float64Elements);
3173           return ElementsHeader::length;
3174       }
3175
3176  @@ -643,7 +643,7 @@ class TypedElementsHeader : public ElementsHeader
3177   template<typename T> inline void
3178   TypedElementsHeader<T>::assign(uint32_t index, double d)
3179   {
3180  -    MOZ_NOT_REACHED("didn't specialize for this element type");
3181  +    TBB_MOZ_NOT_REACHED("didn't specialize for this element type");
3182   }
3183
3184   template<> inline void
3185  @@ -809,84 +809,84 @@ class ArrayBufferElementsHeader : public ElementsHeader
3186   inline DenseElementsHeader &
3187   ElementsHeader::asDenseElements()
3188   {
3189  -    MOZ_ASSERT(isDenseElements());
3190  +    TBB_MOZ_ASSERT(isDenseElements());
3191       return *static_cast<DenseElementsHeader *>(this);
3192   }
3193
3194   inline SparseElementsHeader &
3195   ElementsHeader::asSparseElements()
3196   {
3197  -    MOZ_ASSERT(isSparseElements());
3198  +    TBB_MOZ_ASSERT(isSparseElements());
3199       return *static_cast<SparseElementsHeader *>(this);
3200   }
3201
3202   inline Uint8ElementsHeader &
3203   ElementsHeader::asUint8Elements()
3204   {
3205  -    MOZ_ASSERT(isUint8Elements());
3206  +    TBB_MOZ_ASSERT(isUint8Elements());
3207       return *static_cast<Uint8ElementsHeader *>(this);
3208   }
3209
3210   inline Int8ElementsHeader &
3211   ElementsHeader::asInt8Elements()
3212   {
3213  -    MOZ_ASSERT(isInt8Elements());
3214  +    TBB_MOZ_ASSERT(isInt8Elements());
3215       return *static_cast<Int8ElementsHeader *>(this);
3216   }
3217
3218   inline Uint16ElementsHeader &
```

```
3219  ElementsHeader::asUint16Elements()
3220  {
3221 -    MOZ_ASSERT(isUint16Elements());
3222 +    TBB_MOZ_ASSERT(isUint16Elements());
3223       return *static_cast<Uint16ElementsHeader *>(this);
3224  }
3225
3226  inline Int16ElementsHeader &
3227  ElementsHeader::asInt16Elements()
3228  {
3229 -    MOZ_ASSERT(isInt16Elements());
3230 +    TBB_MOZ_ASSERT(isInt16Elements());
3231       return *static_cast<Int16ElementsHeader *>(this);
3232  }
3233
3234  inline Uint32ElementsHeader &
3235  ElementsHeader::asUint32Elements()
3236  {
3237 -    MOZ_ASSERT(isUint32Elements());
3238 +    TBB_MOZ_ASSERT(isUint32Elements());
3239       return *static_cast<Uint32ElementsHeader *>(this);
3240  }
3241
3242  inline Int32ElementsHeader &
3243  ElementsHeader::asInt32Elements()
3244  {
3245 -    MOZ_ASSERT(isInt32Elements());
3246 +    TBB_MOZ_ASSERT(isInt32Elements());
3247       return *static_cast<Int32ElementsHeader *>(this);
3248  }
3249
3250  inline Uint8ClampedElementsHeader &
3251  ElementsHeader::asUint8ClampedElements()
3252  {
3253 -    MOZ_ASSERT(isUint8ClampedElements());
3254 +    TBB_MOZ_ASSERT(isUint8ClampedElements());
3255       return *static_cast<Uint8ClampedElementsHeader *>(this);
3256  }
3257
3258  inline Float32ElementsHeader &
3259  ElementsHeader::asFloat32Elements()
3260  {
3261 -    MOZ_ASSERT(isFloat32Elements());
3262 +    TBB_MOZ_ASSERT(isFloat32Elements());
3263       return *static_cast<Float32ElementsHeader *>(this);
3264  }
3265
3266  inline Float64ElementsHeader &
3267  ElementsHeader::asFloat64Elements()
3268  {
3269 -    MOZ_ASSERT(isFloat64Elements());
```

```
3270 +    TBB_MOZ_ASSERT(isFloat64Elements());
3271      return *static_cast<Float64ElementsHeader *>(this);
3272  }
3273
3274  inline ArrayBufferElementsHeader &
3275  ElementsHeader::asArrayBufferElements()
3276  {
3277 -    MOZ_ASSERT(isArrayBufferElements());
3278 +    TBB_MOZ_ASSERT(isArrayBufferElements());
3279      return *static_cast<ArrayBufferElementsHeader *>(this);
3280  }
3281
3282 @@ -1021,7 +1021,7 @@ class ObjectElements
3283      uint32_t length;
3284
3285      void staticAsserts() {
3286 -        MOZ_STATIC_ASSERT(sizeof(ObjectElements) == VALUES_PER_HEADER * sizeof(Value
     ),
3287 +        TBB_MOZ_STATIC_ASSERT(sizeof(ObjectElements) == VALUES_PER_HEADER * sizeof(
     Value),
3288                            "Elements size and values-per-Elements mismatch");
3289      }
3290
3291 @@ -1166,18 +1166,18 @@ class ObjectImpl : public gc::Cell
3292
3293    private:
3294      static void staticAsserts() {
3295 -        MOZ_STATIC_ASSERT(sizeof(ObjectImpl) == sizeof(shadow::Object),
3296 +        TBB_MOZ_STATIC_ASSERT(sizeof(ObjectImpl) == sizeof(shadow::Object),
3297                            "shadow interface must match actual implementation");
3298 -        MOZ_STATIC_ASSERT(sizeof(ObjectImpl) % sizeof(Value) == 0,
3299 +        TBB_MOZ_STATIC_ASSERT(sizeof(ObjectImpl) % sizeof(Value) == 0,
3300                            "fixed slots after an object must be aligned");
3301
3302 -        MOZ_STATIC_ASSERT(offsetof(ObjectImpl, shape_) == offsetof(shadow::Object,
     shape),
3303 +        TBB_MOZ_STATIC_ASSERT(offsetof(ObjectImpl, shape_) == offsetof(shadow::
     Object, shape),
3304                            "shadow shape must match actual shape");
3305 -        MOZ_STATIC_ASSERT(offsetof(ObjectImpl, type_) == offsetof(shadow::Object,
     type),
3306 +        TBB_MOZ_STATIC_ASSERT(offsetof(ObjectImpl, type_) == offsetof(shadow::Object
     , type),
3307                            "shadow type must match actual type");
3308 -        MOZ_STATIC_ASSERT(offsetof(ObjectImpl, slots) == offsetof(shadow::Object,
     slots),
3309 +        TBB_MOZ_STATIC_ASSERT(offsetof(ObjectImpl, slots) == offsetof(shadow::Object
     , slots),
3310                            "shadow slots must match actual slots");
3311 -        MOZ_STATIC_ASSERT(offsetof(ObjectImpl, elements) == offsetof(shadow::Object,
      _1),
```

```
3312 +        TBB_MOZ_STATIC_ASSERT(offsetof(ObjectImpl, elements) == offsetof(shadow::
     Object, _1),
3313                              "shadow placeholder must match actual elements");
3314      }
3315
3316 @@ -1213,7 +1213,7 @@ class ObjectImpl : public gc::Cell
3317      bool makeElementsSparse(JSContext *cx) {
3318          NEW_OBJECT_REPRESENTATION_ONLY();
3319
3320 -        MOZ_NOT_REACHED("NYI");
3321 +        TBB_MOZ_NOT_REACHED("NYI");
3322          return false;
3323      }
3324
3325 @@ -1272,7 +1272,7 @@ class ObjectImpl : public gc::Cell
3326       */
3327      void copySlotRange(uint32_t start, const Value *vector, uint32_t length);
3328
3329 -#ifdef DEBUG
3330 +#ifndef TOR_NASSERT
3331      enum SentinelAllowed {
3332          SENTINEL_NOT_ALLOWED,
3333          SENTINEL_ALLOWED
3334 @@ -1307,7 +1307,7 @@ class ObjectImpl : public gc::Cell
3335      {
3336          NEW_OBJECT_REPRESENTATION_ONLY();
3337
3338 -        MOZ_NOT_REACHED("NYI");
3339 +        TBB_MOZ_NOT_REACHED("NYI");
3340          return Failure;
3341      }
3342
3343 @@ -1320,7 +1320,7 @@ class ObjectImpl : public gc::Cell
3344      inline js::TaggedProto getTaggedProto() const;
3345
3346      Shape * lastProperty() const {
3347 -        MOZ_ASSERT(shape_);
3348 +        TBB_MOZ_ASSERT(shape_);
3349          return shape_;
3350      }
3351
3352 @@ -1333,7 +1333,7 @@ class ObjectImpl : public gc::Cell
3353      inline bool isNative() const;
3354
3355      types::TypeObject *type() const {
3356 -        MOZ_ASSERT(!hasLazyType());
3357 +        TBB_MOZ_ASSERT(!hasLazyType());
3358          return type_;
3359      }
3360
3361 @@ -1403,7 +1403,7 @@ class ObjectImpl : public gc::Cell
```

```
3362      inline bool inDictionaryMode() const;
3363
3364      const Value &getSlot(uint32_t slot) const {
3365  -         MOZ_ASSERT(slotInRange(slot));
3366  +         TBB_MOZ_ASSERT(slotInRange(slot));
3367            uint32_t fixed = numFixedSlots();
3368            if (slot < fixed)
3369                return fixedSlots()[slot];
3370  @@ -1423,12 +1423,12 @@ class ObjectImpl : public gc::Cell
3371             * object, which may be necessary when fetching zero-length arrays of
3372             * slots (e.g. for callObjVarArray).
3373             */
3374  -         MOZ_ASSERT(slotInRange(slot, SENTINEL_ALLOWED));
3375  +         TBB_MOZ_ASSERT(slotInRange(slot, SENTINEL_ALLOWED));
3376            return getSlotAddressUnchecked(slot);
3377        }
3378
3379        HeapSlot &getSlotRef(uint32_t slot) {
3380  -         MOZ_ASSERT(slotInRange(slot));
3381  +         TBB_MOZ_ASSERT(slotInRange(slot));
3382            return *getSlotAddress(slot);
3383        }
3384
3385  @@ -1444,12 +1444,12 @@ class ObjectImpl : public gc::Cell
3386        /* For slots which are known to always be fixed, due to the way they are
              allocated. */
3387
3388        HeapSlot &getFixedSlotRef(uint32_t slot) {
3389  -         MOZ_ASSERT(slot < numFixedSlots());
3390  +         TBB_MOZ_ASSERT(slot < numFixedSlots());
3391            return fixedSlots()[slot];
3392        }
3393
3394        const Value &getFixedSlot(uint32_t slot) const {
3395  -         MOZ_ASSERT(slot < numFixedSlots());
3396  +         TBB_MOZ_ASSERT(slot < numFixedSlots());
3397            return fixedSlots()[slot];
3398        }
3399
3400  @@ -1479,7 +1479,7 @@ class ObjectImpl : public gc::Cell
3401        }
3402
3403        inline HeapSlot *fixedElements() const {
3404  -         MOZ_STATIC_ASSERT(2 * sizeof(Value) == sizeof(ObjectElements),
3405  +         TBB_MOZ_STATIC_ASSERT(2 * sizeof(Value) == sizeof(ObjectElements),
3406                               "when elements are stored inline, the first two "
3407                               "slots will hold the ObjectElements header");
3408            return &fixedSlots()[2];
3409  @@ -1524,8 +1524,8 @@ class ObjectImpl : public gc::Cell
3410             * Private pointers are stored immediately after the last fixed slot of
3411             * the object.
```

```
3412           */
3413 -         MOZ_ASSERT(nfixed == numFixedSlots());
3414 -         MOZ_ASSERT(hasPrivate());
3415 +         TBB_MOZ_ASSERT(nfixed == numFixedSlots());
3416 +         TBB_MOZ_ASSERT(hasPrivate());
3417           HeapSlot *end = &fixedSlots()[nfixed];
3418           return *reinterpret_cast<void**>(end);
3419       }
3420 diff --git a/js/src/vm/SPSProfiler.cpp b/js/src/vm/SPSProfiler.cpp
3421 index ec3e5fb..9781c53 100644
3422 --- a/js/src/vm/SPSProfiler.cpp
3423 +++ b/js/src/vm/SPSProfiler.cpp
3424 @@ -4,7 +4,7 @@
3425    * License, v. 2.0. If a copy of the MPL was not distributed with this
3426    * file, You can obtain one at http://mozilla.org/MPL/2.0/. */
3427
3428 -#include "mozilla/DebugOnly.h"
3429 +#include "mozilla/DebugOnlyTor.h"
3430
3431  #include "jsnum.h"
3432  #include "jsscript.h"
3433 @@ -16,7 +16,7 @@
3434
3435  using namespace js;
3436
3437 -using mozilla::DebugOnly;
3438 +using mozilla::DebugOnlyTor;
3439
3440  SPSProfiler::SPSProfiler(JSRuntime *rt)
3441    : rt(rt),
3442 @@ -205,7 +205,7 @@ SPSProfiler::pop()
3443  const char*
3444  SPSProfiler::allocProfileString(JSContext *cx, JSScript *script, JSFunction *
       maybeFun)
3445  {
3446 -    DebugOnly<uint64_t> gcBefore = cx->runtime()->gcNumber;
3447 +    DebugOnlyTor<uint64_t> gcBefore = cx->runtime()->gcNumber;
3448      StringBuffer buf(cx);
3449      bool hasAtom = maybeFun != NULL && maybeFun->displayAtom() != NULL;
3450      if (hasAtom) {
3451 diff --git a/js/src/vm/SPSProfiler.h b/js/src/vm/SPSProfiler.h
3452 index f9b426e..2f3e00c 100644
3453 --- a/js/src/vm/SPSProfiler.h
3454 +++ b/js/src/vm/SPSProfiler.h
3455 @@ -9,7 +9,7 @@
3456
3457  #include <stddef.h>
3458
3459 -#include "mozilla/DebugOnly.h"
3460 +#include "mozilla/DebugOnlyTor.h"
3461  #include "mozilla/GuardObjects.h"
```

```
 #include "mozilla/HashFunctions.h"

@@ -210,7 +210,7 @@ class SPSEntryMarker

   private:
     SPSProfiler *profiler;
-    mozilla::DebugOnly<uint32_t> size_before;
+    mozilla::DebugOnlyTor<uint32_t> size_before;
     MOZ_DECL_USE_GUARD_OBJECT_NOTIFIER
 };

diff --git a/js/src/vm/Shape.cpp b/js/src/vm/Shape.cpp
index da08e89..76ce1f7 100644
--- a/js/src/vm/Shape.cpp
+++ b/js/src/vm/Shape.cpp
@@ -6,7 +6,7 @@

 /* JS symbol tables. */

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"
 #include "mozilla/PodOperations.h"

 #include "jsapi.h"
@@ -25,7 +25,7 @@
 using namespace js;
 using namespace js::gc;

-using mozilla::DebugOnly;
+using mozilla::DebugOnlyTor;
 using mozilla::PodZero;

 bool
@@ -163,7 +163,7 @@ ShapeTable::search(jsid id, bool adding)
     hash2 = HASH2(hash0, sizeLog2, hashShift);
     sizeMask = JS_BITMASK(sizeLog2);

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     uintptr_t collision_flag = SHAPE_COLLISION;
 #endif

@@ -174,7 +174,7 @@ ShapeTable::search(jsid id, bool adding)
         firstRemoved = NULL;
         if (adding && !SHAPE_HAD_COLLISION(stored))
             SHAPE_FLAG_COLLISION(spp, shape);
-#ifdef DEBUG
+#ifndef TOR_NASSERT
         collision_flag &= uintptr_t(*spp) & SHAPE_COLLISION;
 #endif
     }
```

```
@@ -200,7 +200,7 @@ ShapeTable::search(jsid id, bool adding)
         } else {
             if (adding && !SHAPE_HAD_COLLISION(stored))
                 SHAPE_FLAG_COLLISION(spp, shape);
-#ifdef DEBUG
+#ifndef TOR_NASSERT
             collision_flag &= uintptr_t(*spp) & SHAPE_COLLISION;
 #endif
         }
@@ -1450,8 +1450,8 @@ JSCompartment::sweepInitialShapeTable()
             if (IsShapeAboutToBeFinalized(&shape) || (entry.proto.isObject() &&
                 IsObjectAboutToBeFinalized(&proto))) {
                 e.removeFront();
             } else {
-#ifdef DEBUG
-                DebugOnly<JSObject *> parent = shape->getObjectParent();
+#ifndef TOR_NASSERT
+                DebugOnlyTor<JSObject *> parent = shape->getObjectParent();
                 JS_ASSERT(!parent || !IsObjectAboutToBeFinalized(&parent));
                 JS_ASSERT(parent == shape->getObjectParent());
 #endif
diff --git a/js/src/vm/Stack-inl.h b/js/src/vm/Stack-inl.h
index db6fc22..a035acb 100644
--- a/js/src/vm/Stack-inl.h
+++ b/js/src/vm/Stack-inl.h
@@ -849,7 +849,7 @@ InterpreterActivation::InterpreterActivation(JSContext *cx,
     StackFrame *entry, F
     entry_(entry),
     current_(entry),
     regs_(regs)
-#ifdef DEBUG
+#ifndef TOR_NASSERT
   , oldFrameCount_(cx_->runtime()->interpreterStack().frameCount_)
 #endif
 {}
diff --git a/js/src/vm/Stack.h b/js/src/vm/Stack.h
index fffcf73..46f90a8 100644
--- a/js/src/vm/Stack.h
+++ b/js/src/vm/Stack.h
@@ -1217,7 +1217,7 @@ class InterpreterActivation : public Activation
     StackFrame *current_;     // The most recent frame.
     FrameRegs &regs_;

-#ifdef DEBUG
+#ifndef TOR_NASSERT
     size_t oldFrameCount_;
 #endif

diff --git a/js/src/vm/StringBuffer.h b/js/src/vm/StringBuffer.h
index 9c40fec..587537b 100644
--- a/js/src/vm/StringBuffer.h
```

```
+++ b/js/src/vm/StringBuffer.h
@@ -7,7 +7,7 @@
 #ifndef vm_StringBuffer_h
 #define vm_StringBuffer_h

-#include "mozilla/DebugOnly.h"
+#include "mozilla/DebugOnlyTor.h"

 #include "jscntxt.h"

@@ -120,8 +120,8 @@ StringBuffer::appendInflated(const char *cstr, size_t cstrlen)
     size_t lengthBefore = length();
     if (!cb.growByUninitialized(cstrlen))
         return false;
-    mozilla::DebugOnly<size_t> oldcstrlen = cstrlen;
-    mozilla::DebugOnly<bool> ok = InflateStringToBuffer(context(), cstr, cstrlen,
+    mozilla::DebugOnlyTor<size_t> oldcstrlen = cstrlen;
+    mozilla::DebugOnlyTor<bool> ok = InflateStringToBuffer(context(), cstr, cstrlen,
                                                         begin() + lengthBefore, &
                                                             cstrlen);
     JS_ASSERT(ok && oldcstrlen == cstrlen);
     return true;
diff --git a/media/libnestegg/src/halloc.c b/media/libnestegg/src/halloc.c
index 5382c56..962f20d 100644
--- a/media/libnestegg/src/halloc.c
+++ b/media/libnestegg/src/halloc.c
@@ -75,7 +75,7 @@ void * halloc(void * ptr, size_t len)
    p = allocator(0, len + sizeof_hblock);
    if (! p)
      return NULL;
-#ifndef NDEBUG
+#ifndef TOR_NASSERT
    p->magic = HH_MAGIC;
 #endif
    hlist_init(&p->children);
@@ -236,7 +236,7 @@ static void _free_children(hblock_t * p)
 {
  hlist_item_t * i, * tmp;

-#ifndef NDEBUG
+#ifndef TOR_NASSERT
  /*
   *  this catches loops in hierarchy with almost zero
   *  overhead (compared to _relate() running time)
diff --git a/mfbt/AssertionsTor.h b/mfbt/AssertionsTor.h
new file mode 100644
index 0000000..0e8ea18
--- /dev/null
+++ b/mfbt/AssertionsTor.h
@@ -0,0 +1,436 @@
+/* -*- Mode: C++; tab-width: 2; indent-tabs-mode: nil; c-basic-offset: 2 -*- */
```

```
+/* This Source Code Form is subject to the terms of the Mozilla Public
+ * License, v. 2.0. If a copy of the MPL was not distributed with this
+ * file, You can obtain one at http://mozilla.org/MPL/2.0/. */
+
+/* Implementations of runtime and static assertion macros for C and C++. */
+
+#ifndef tbb_Assertions_h_
+#define tbb_Assertions_h_
+
+#include "mozilla/Attributes.h"
+#include "mozilla/Compiler.h"
+#include "mozilla/Likely.h"
+
+#include <stddef.h>
+#include <stdio.h>
+#include <stdlib.h>
+#ifdef WIN32
+   /*
+    * TerminateProcess and GetCurrentProcess are defined in <winbase.h>, which
+    * further depends on <windef.h>.  We hardcode these few definitions manually
+    * because those headers clutter the global namespace with a significant
+    * number of undesired macros and symbols.
+    */
+#  ifdef __cplusplus
+   extern "C" {
+#  endif
+   __declspec(dllimport) int __stdcall
+   TerminateProcess(void* hProcess, unsigned int uExitCode);
+   __declspec(dllimport) void* __stdcall GetCurrentProcess(void);
+#  ifdef __cplusplus
+   }
+#  endif
+#else
+#  include <signal.h>
+#endif
+#ifdef ANDROID
+#  include <android/log.h>
+#endif
+
+/*
+ * TBB_MOZ_STATIC_ASSERT may be used to assert a condition *at compile time*.  This
+ * can be useful when you make certain assumptions about what must hold for
+ * optimal, or even correct, behavior.  For example, you might assert that the
+ * size of a struct is a multiple of the target architecture's word size:
+ *
+ *   struct S { ... };
+ *   TBB_MOZ_STATIC_ASSERT(sizeof(S) % sizeof(size_t) == 0,
+ *                     "S should be a multiple of word size for efficiency");
+ *
+ * This macro can be used in any location where both an extern declaration and a
+ * typedef could be used.
```

```
3663 + *
3664 + * Be aware of the gcc 4.2 concerns noted further down when writing patches that
3665 + * use this macro, particularly if a patch only bounces on OS X.
3666 + */
3667 +#ifdef __cplusplus
3668 +#  if defined(__clang__)
3669 +#    ifndef __has_extension
3670 +#      define __has_extension __has_feature /* compatibility, for older versions of
         clang */
3671 +#    endif
3672 +#    if __has_extension(cxx_static_assert)
3673 +#      define TBB_MOZ_STATIC_ASSERT(cond, reason)    static_assert((cond), reason)
3674 +#    endif
3675 +#  elif defined(__GNUC__)
3676 +#    if (defined(__GXX_EXPERIMENTAL_CXX0X__) || __cplusplus >= 201103L)
3677 +#      define TBB_MOZ_STATIC_ASSERT(cond, reason)    static_assert((cond), reason)
3678 +#    endif
3679 +#  elif defined(_MSC_VER)
3680 +#    if _MSC_VER >= 1600 /* MSVC 10 */
3681 +#      define TBB_MOZ_STATIC_ASSERT(cond, reason)    static_assert((cond), reason)
3682 +#    endif
3683 +#  elif defined(__HP_aCC)
3684 +#    if __HP_aCC >= 62500 && defined(_HP_CXX0x_SOURCE)
3685 +#      define TBB_MOZ_STATIC_ASSERT(cond, reason)    static_assert((cond), reason)
3686 +#    endif
3687 +#  endif
3688 +#endif
3689 +#ifndef TBB_MOZ_STATIC_ASSERT
3690 +   /*
3691 +    * Some of the definitions below create an otherwise-unused typedef.  This
3692 +    * triggers compiler warnings with some versions of gcc, so mark the typedefs
3693 +    * as permissibly-unused to disable the warnings.
3694 +    */
3695 +#  if defined(__GNUC__)
3696 +#    define TBB_MOZ_STATIC_ASSERT_UNUSED_ATTRIBUTE __attribute__((unused))
3697 +#  else
3698 +#    define TBB_MOZ_STATIC_ASSERT_UNUSED_ATTRIBUTE /* nothing */
3699 +#  endif
3700 +#  define TBB_MOZ_STATIC_ASSERT_GLUE1(x, y)          x##y
3701 +#  define TBB_MOZ_STATIC_ASSERT_GLUE(x, y)           TBB_MOZ_STATIC_ASSERT_GLUE1(x,
         y)
3702 +#  if defined(__SUNPRO_CC)
3703 +     /*
3704 +      * The Sun Studio C++ compiler is buggy when declaring, inside a function,
3705 +      * another extern'd function with an array argument whose length contains a
3706 +      * sizeof, triggering the error message "sizeof expression not accepted as
3707 +      * size of array parameter".  This bug (6688515, not public yet) would hit
3708 +      * defining moz_static_assert as a function, so we always define an extern
3709 +      * array for Sun Studio.
3710 +      *
3711 +      * We include the line number in the symbol name in a best-effort attempt
```

```
+      * to avoid conflicts (see below).
+      */
+#    define TBB_MOZ_STATIC_ASSERT(cond, reason) \
+       extern char TBB_MOZ_STATIC_ASSERT_GLUE(moz_static_assert, __LINE__)[(cond) ? 1 : -1]
+#  elif defined(__COUNTER__)
+     /*
+      * If there was no preferred alternative, use a compiler-agnostic version.
+      *
+      * Note that the non-__COUNTER__ version has a bug in C++: it can't be used
+      * in both |extern "C"| and normal C++ in the same translation unit.  (Alas
+      * |extern "C"| isn't allowed in a function.)  The only affected compiler
+      * we really care about is gcc 4.2.  For that compiler and others like it,
+      * we include the line number in the function name to do the best we can to
+      * avoid conflicts.  These should be rare: a conflict would require use of
+      * TBB_MOZ_STATIC_ASSERT on the same line in separate files in the same
+      * translation unit, *and* the uses would have to be in code with
+      * different linkage, *and* the first observed use must be in C++-linkage
+      * code.
+      */
+#    define TBB_MOZ_STATIC_ASSERT(cond, reason) \
+       typedef int TBB_MOZ_STATIC_ASSERT_GLUE(moz_static_assert, __COUNTER__)[(cond) ? 1 : -1] TBB_MOZ_STATIC_ASSERT_UNUSED_ATTRIBUTE
+#  else
+#    define TBB_MOZ_STATIC_ASSERT(cond, reason) \
+       extern void TBB_MOZ_STATIC_ASSERT_GLUE(moz_static_assert, __LINE__)(int arg[(cond) ? 1 : -1]) TBB_MOZ_STATIC_ASSERT_UNUSED_ATTRIBUTE
+#  endif
+#endif
+
+#define TBB_MOZ_STATIC_ASSERT_IF(cond, expr, reason)  TBB_MOZ_STATIC_ASSERT(!(cond) || (expr), reason)
+
+#ifdef __cplusplus
+extern "C" {
+#endif
+
+/*
+ * Prints |s| as an assertion failure (using file and ln as the location of the
+ * assertion) to the standard debug-output channel.
+ *
+ * Usually you should use TBB_MOZ_ASSERT or TBB_MOZ_CRASH instead of this method.  This
+ * method is primarily for internal use in this header, and only secondarily
+ * for use in implementing release-build assertions.
+ */
+static MOZ_ALWAYS_INLINE void
+TBB_MOZ_ReportAssertionFailure(const char* s, const char* file, int ln)
+{
+#ifdef ANDROID
+  __android_log_print(ANDROID_LOG_FATAL, "TBB_MOZ_Assert",
```

```
+                     "Assertion failure: %s, at %s:%d\n", s, file, ln);
+#else
+  fprintf(stderr, "Assertion failure: %s, at %s:%d\n", s, file, ln);
+  fflush(stderr);
+#endif
+}
+
+static MOZ_ALWAYS_INLINE void
+TBB_MOZ_ReportCrash(const char* s, const char* file, int ln)
+{
+#ifdef ANDROID
+    __android_log_print(ANDROID_LOG_FATAL, "TBB_MOZ_CRASH",
+                        "Hit TBB_MOZ_CRASH(%s) at %s:%d\n", s, file, ln);
+#else
+  fprintf(stderr, "Hit TBB_MOZ_CRASH(%s) at %s:%d\n", s, file, ln);
+  fflush(stderr);
+#endif
+}
+
+/**
+ * TBB_MOZ_REALLY_CRASH is used in the implementation of TBB_MOZ_CRASH().  You
    should
+ * call TBB_MOZ_CRASH instead.
+ */
+#if defined(_MSC_VER)
+   /*
+    * On MSVC use the __debugbreak compiler intrinsic, which produces an inline
+    * (not nested in a system function) breakpoint.  This distinctively invokes
+    * Breakpad without requiring system library symbols on all stack-processing
+    * machines, as a nested breakpoint would require.
+    *
+    * We use TerminateProcess with the exit code aborting would generate
+    * because we don't want to invoke atexit handlers, destructors, library
+    * unload handlers, and so on when our process might be in a compromised
+    * state.
+    *
+    * We don't use abort() because it'd cause Windows to annoyingly pop up the
+    * process error dialog multiple times.  See bug 345118 and bug 426163.
+    *
+    * We follow TerminateProcess() with a call to TBB_MOZ_NoReturn() so that the
+    * compiler doesn't hassle us to provide a return statement after a
+    * TBB_MOZ_REALLY_CRASH() call.
+    *
+    * (Technically these are Windows requirements, not MSVC requirements.  But
+    * practically you need MSVC for debugging, and we only ship builds created
+    * by MSVC, so doing it this way reduces complexity.)
+    */
+
+__declspec(noreturn) __inline void TBB_MOZ_NoReturn() {}
+
+#  ifdef __cplusplus
```

```
+#    define TBB_MOZ_REALLY_CRASH() \
+       do { \
+         __debugbreak(); \
+         *((volatile int*) NULL) = 123; \
+         ::TerminateProcess(::GetCurrentProcess(), 3); \
+         ::TBB_MOZ_NoReturn(); \
+       } while (0)
+#  else
+#    define TBB_MOZ_REALLY_CRASH() \
+       do { \
+         __debugbreak(); \
+         *((volatile int*) NULL) = 123; \
+         TerminateProcess(GetCurrentProcess(), 3); \
+         TBB_MOZ_NoReturn(); \
+       } while (0)
+#  endif
+#else
+#  ifdef __cplusplus
+#    define TBB_MOZ_REALLY_CRASH() \
+       do { \
+         *((volatile int*) NULL) = 123; \
+         ::abort(); \
+       } while (0)
+#  else
+#    define TBB_MOZ_REALLY_CRASH() \
+       do { \
+         *((volatile int*) NULL) = 123; \
+         abort(); \
+       } while (0)
+#  endif
+#endif
+
+/*
+ * TBB_MOZ_CRASH([explanation-string]) crashes the program, plain and simple, in a
+ * Breakpad-compatible way, in both debug and release builds.
+ *
+ * TBB_MOZ_CRASH is a good solution for "handling" failure cases when you're
+ * unwilling or unable to handle them more cleanly -- for OOM, for likely memory
+ * corruption, and so on.  It's also a good solution if you need safe behavior
+ * in release builds as well as debug builds.  But if the failure is one that
+ * should be debugged and fixed, TBB_MOZ_ASSERT is generally preferable.
+ *
+ * The optional explanation-string, if provided, must be a string literal
+ * explaining why we're crashing.  This argument is intended for use with
+ * TBB_MOZ_CRASH() calls whose rationale is non-obvious; don't use it if it's
+ * obvious why we're crashing.
+ *
+ * If we're a DEBUG build and we crash at a TBB_MOZ_CRASH which provides an
+ * explanation-string, we print the string to stderr.  Otherwise, we don't
+ * print anything; this is because we want TBB_MOZ_CRASH to be 100% safe in release
+ * builds, and it's hard to print to stderr safely when memory might have been
```

```
+ * corrupted.
+ */
+#ifdef TOR_NASSERT
+#  define TBB_MOZ_CRASH(...) TBB_MOZ_REALLY_CRASH()
+#else
+#  define TBB_MOZ_CRASH(...) \
+     do { \
+       TBB_MOZ_ReportCrash("" __VA_ARGS__, __FILE__, __LINE__); \
+       TBB_MOZ_REALLY_CRASH(); \
+     } while(0)
+#endif
+
+#ifdef __cplusplus
+} /* extern "C" */
+#endif
+
+/*
+ * TBB_MOZ_ASSERT(expr [, explanation-string]) asserts that |expr| must be truthy in
+ * debug builds.  If it is, execution continues.  Otherwise, an error message
+ * including the expression and the explanation-string (if provided) is printed,
+ * an attempt is made to invoke any existing debugger, and execution halts.
+ * TBB_MOZ_ASSERT is fatal: no recovery is possible.  Do not assert a condition
+ * which can correctly be falsy.
+ *
+ * The optional explanation-string, if provided, must be a string literal
+ * explaining the assertion.  It is intended for use with assertions whose
+ * correctness or rationale is non-obvious, and for assertions where the "real"
+ * condition being tested is best described prosaically.  Don't provide an
+ * explanation if it's not actually helpful.
+ *
+ *   // No explanation needed: pointer arguments often must not be NULL.
+ *   TBB_MOZ_ASSERT(arg);
+ *
+ *   // An explanation can be helpful to explain exactly how we know an
+ *   // assertion is valid.
+ *   TBB_MOZ_ASSERT(state == WAITING_FOR_RESPONSE,
+ *                  "given that <thingA> and <thingB>, we must have...");
+ *
+ *   // Or it might disambiguate multiple identical (save for their location)
+ *   // assertions of the same expression.
+ *   TBB_MOZ_ASSERT(getSlot(PRIMITIVE_THIS_SLOT).isUndefined(),
+ *                  "we already set [[PrimitiveThis]] for this Boolean object");
+ *   TBB_MOZ_ASSERT(getSlot(PRIMITIVE_THIS_SLOT).isUndefined(),
+ *                  "we already set [[PrimitiveThis]] for this String object");
+ *
+ * TBB_MOZ_ASSERT has no effect in non-debug builds.  It is designed to catch bugs
+ * *only* during debugging, not "in the field".
+ */
+#ifndef TOR_NASSERT
+   /* First the single-argument form. */
+#  define TBB_MOZ_ASSERT_HELPER1(expr) \
```

```
+     do { \
+       if (MOZ_UNLIKELY(!(expr))) { \
+         TBB_MOZ_ReportAssertionFailure(#expr, __FILE__, __LINE__); \
+         TBB_MOZ_REALLY_CRASH(); \
+       } \
+     } while (0)
+   /* Now the two-argument form. */
+#  define TBB_MOZ_ASSERT_HELPER2(expr, explain) \
+     do { \
+       if (MOZ_UNLIKELY(!(expr))) { \
+         TBB_MOZ_ReportAssertionFailure(#expr " (" explain ")", __FILE__, __LINE__);
      \
+         TBB_MOZ_REALLY_CRASH(); \
+       } \
+     } while (0)
+   /* And now, helper macrology up the wazoo. */
+   /*
+    * Count the number of arguments passed to TBB_MOZ_ASSERT, very carefully
+    * tiptoeing around an MSVC bug where it improperly expands __VA_ARGS__ as a
+    * single token in argument lists.  See these URLs for details:
+    *
+    *   http://connect.microsoft.com/VisualStudio/feedback/details/380090/variadic-
     macro-replacement
+    *   http://cplusplus.co.il/2010/07/17/variadic-macro-to-count-number-of-
     arguments/#comment-644
+    */
+#  define TBB_MOZ_COUNT_ASSERT_ARGS_IMPL2(_1, _2, count, ...) \
+     count
+#  define TBB_MOZ_COUNT_ASSERT_ARGS_IMPL(args) \
+  TBB_MOZ_COUNT_ASSERT_ARGS_IMPL2 args
+#  define TBB_MOZ_COUNT_ASSERT_ARGS(...) \
+     TBB_MOZ_COUNT_ASSERT_ARGS_IMPL((__VA_ARGS__, 2, 1, 0))
+   /* Pick the right helper macro to invoke. */
+#  define TBB_MOZ_ASSERT_CHOOSE_HELPER2(count) TBB_MOZ_ASSERT_HELPER##count
+#  define TBB_MOZ_ASSERT_CHOOSE_HELPER1(count) TBB_MOZ_ASSERT_CHOOSE_HELPER2(count)
+#  define TBB_MOZ_ASSERT_CHOOSE_HELPER(count) TBB_MOZ_ASSERT_CHOOSE_HELPER1(count)
+   /* The actual macro. */
+#  define TBB_MOZ_ASSERT_GLUE(x, y) x y
+#  define TBB_MOZ_ASSERT(...) \
+     TBB_MOZ_ASSERT_GLUE(TBB_MOZ_ASSERT_CHOOSE_HELPER(TBB_MOZ_COUNT_ASSERT_ARGS(
     __VA_ARGS__)), \
+                         (__VA_ARGS__))
+#else
+#  define TBB_MOZ_ASSERT(...) do { } while(0)
+#endif /* !TOR_NASSERT */
+
+/*
+ * TBB_MOZ_ASSERT_IF(cond1, cond2) is equivalent to TBB_MOZ_ASSERT(cond2) if cond1
     is
+ * true.
+ *
```

```
+ *   TBB_MOZ_ASSERT_IF(isPrime(num), num == 2 || isOdd(num));
+ *
+ * As with TBB_MOZ_ASSERT, TBB_MOZ_ASSERT_IF has effect only in debug builds.  It is
+ * designed to catch bugs during debugging, not "in the field".
+ */
+#ifndef TOR_NASSERT
+#  define TBB_MOZ_ASSERT_IF(cond, expr) \
+     do { \
+       if (cond) \
+         TBB_MOZ_ASSERT(expr); \
+     } while (0)
+#else
+#  define TBB_MOZ_ASSERT_IF(cond, expr)  do { } while (0)
+#endif
+
+/*
+ * TBB_MOZ_NOT_REACHED_MARKER() expands to an expression which states that it is
+ * undefined behavior for execution to reach this point.  No guarantees are made
+ * about what will happen if this is reached at runtime.  Most code should
+ * probably use the higher level TBB_MOZ_NOT_REACHED, which uses this when
+ * appropriate.
+ */
+#if defined(__clang__)
+#  define TBB_MOZ_NOT_REACHED_MARKER() __builtin_unreachable()
+#elif defined(__GNUC__)
+   /*
+    * __builtin_unreachable() was implemented in gcc 4.5.  If we don't have
+    * that, call a noreturn function; abort() will do nicely.  Qualify the call
+    * in C++ in case there's another abort() visible in local scope.
+    */
+#  if MOZ_GCC_VERSION_AT_LEAST(4, 5, 0)
+#    define TBB_MOZ_NOT_REACHED_MARKER() __builtin_unreachable()
+#  else
+#    ifdef __cplusplus
+#      define TBB_MOZ_NOT_REACHED_MARKER() ::abort()
+#    else
+#      define TBB_MOZ_NOT_REACHED_MARKER() abort()
+#    endif
+#  endif
+#elif defined(_MSC_VER)
+#  define TBB_MOZ_NOT_REACHED_MARKER() __assume(0)
+#else
+#  ifdef __cplusplus
+#    define TBB_MOZ_NOT_REACHED_MARKER() ::abort()
+#  else
+#    define TBB_MOZ_NOT_REACHED_MARKER() abort()
+#  endif
+#endif
+
+/*
+ * TBB_MOZ_NOT_REACHED(reason) indicates that the given point can't be reached
```

```
+ * during execution: simply reaching that point in execution is a bug.  It takes
+ * as an argument an error message indicating the reason why that point should
+ * not have been reachable.
+ *
+ *   // ...in a language parser...
+ *   void handle(BooleanLiteralNode node)
+ *   {
+ *     if (node.isTrue())
+ *       handleTrueLiteral();
+ *     else if (node.isFalse())
+ *       handleFalseLiteral();
+ *     else
+ *       TBB_MOZ_NOT_REACHED("boolean literal that's not true or false?");
+ *   }
+ */
+#if !defined(TOR_NASSERT)
+#  define TBB_MOZ_NOT_REACHED(reason) \
+     do { \
+       TBB_MOZ_ASSERT(false, reason); \
+       TBB_MOZ_NOT_REACHED_MARKER(); \
+     } while (0)
+#else
+#  define TBB_MOZ_NOT_REACHED(reason)  TBB_MOZ_NOT_REACHED_MARKER()
+#endif
+
+/*
+ * TBB_MOZ_ALWAYS_TRUE(expr) and TBB_MOZ_ALWAYS_FALSE(expr) always evaluate the
    provided
+ * expression, in debug builds and in release builds both.  Then, in debug
+ * builds only, the value of the expression is asserted either true or false
+ * using TBB_MOZ_ASSERT.
+ */
+#ifndef TOR_NASSERT
+#  define TBB_MOZ_ALWAYS_TRUE(expr)      TBB_MOZ_ASSERT((expr))
+#  define TBB_MOZ_ALWAYS_FALSE(expr)     TBB_MOZ_ASSERT(!(expr))
+#else
+#  define TBB_MOZ_ALWAYS_TRUE(expr)      ((void)(expr))
+#  define TBB_MOZ_ALWAYS_FALSE(expr)     ((void)(expr))
+#endif
+
+#endif /* mozilla_Assertions_h_ */
diff --git a/mfbt/DebugOnlyTor.h b/mfbt/DebugOnlyTor.h
new file mode 100644
index 0000000..322cb85
--- /dev/null
+++ b/mfbt/DebugOnlyTor.h
@@ -0,0 +1,77 @@
+/* -*- Mode: C++; tab-width: 2; indent-tabs-mode: nil; c-basic-offset: 2 -*- */
+/* This Source Code Form is subject to the terms of the Mozilla Public
+ * License, v. 2.0. If a copy of the MPL was not distributed with this
+ * file, You can obtain one at http://mozilla.org/MPL/2.0/. */
```

```
+
+/*
+ * Provides DebugOnlyTor, a type for variables used only in debug builds (i.e. by
+ * assertions).
+ */
+
+#ifndef tor_DebugOnly_h_
+#define tor_DebugOnly_h_
+
+namespace mozilla {
+
+/**
+ * DebugOnlyTor contains a value of type T, but only in debug builds.  In release
+ * builds, it does not contain a value.  This helper is intended to be used with
+ * MOZ_ASSERT()-style macros, allowing one to write:
+ *
+ *   DebugOnlyTor<bool> check = func();
+ *   MOZ_ASSERT(check);
+ *
+ * more concisely than declaring |check| conditional on #ifdef DEBUG, but also
+ * without allocating storage space for |check| in release builds.
+ *
+ * DebugOnlyTor instances can only be coerced to T in debug builds.  In release
+ * builds they don't have a value, so type coercion is not well defined.
+ */
+template<typename T>
+class DebugOnlyTor
+{
+  public:
+#ifndef TOR_NASSERT
+    T value;
+
+    DebugOnlyTor() { }
+    DebugOnlyTor(const T& other) : value(other) { }
+    DebugOnlyTor(const DebugOnlyTor& other) : value(other.value) { }
+    DebugOnlyTor& operator=(const T& rhs) {
+      value = rhs;
+      return *this;
+    }
+    void operator++(int) {
+      value++;
+    }
+    void operator--(int) {
+      value--;
+    }
+
+    T* operator&() { return &value; }
+
+    operator T&() { return value; }
+    operator const T&() const { return value; }
+
```

```
+    T& operator->() { return value; }
+
+#else
+    DebugOnlyTor() { }
+    DebugOnlyTor(const T&) { }
+    DebugOnlyTor(const DebugOnlyTor&) { }
+    DebugOnlyTor& operator=(const T&) { return *this; }
+    void operator++(int) { }
+    void operator--(int) { }
+#endif
+
+    /*
+     * DebugOnlyTor must always have a destructor or else it will
+     * generate "unused variable" warnings, exactly what it's intended
+     * to avoid!
+     */
+    ~DebugOnlyTor() {}
+};
+
+}
+
+#endif  /* tor_DebugOnly_h_ */
diff --git a/mfbt/exported_headers.mk b/mfbt/exported_headers.mk
index 6370936..5582fcd 100644
--- a/mfbt/exported_headers.mk
+++ b/mfbt/exported_headers.mk
@@ -10,6 +10,7 @@ EXPORTS_NAMESPACES += mozilla

 EXPORTS_mozilla += \
   Assertions.h \
+  AssertionsTor.h \
   Atomics.h \
   Attributes.h \
   BloomFilter.h \
@@ -19,6 +20,7 @@ EXPORTS_mozilla += \
   Compiler.h \
   Constants.h \
   DebugOnly.h \
+  DebugOnlyTor.h \
   decimal/Decimal.h \
   Endian.h \
   EnumSet.h \
diff --git a/xpcom/base/nsAutoPtr.h b/xpcom/base/nsAutoPtr.h
index e33eaeb..009ef8b 100644
--- a/xpcom/base/nsAutoPtr.h
+++ b/xpcom/base/nsAutoPtr.h
@@ -994,7 +994,7 @@ class nsRefPtr
           // parameters where rhs bay be a T** or an I** where I is a base class
           // of T.
         {
-          NS_ASSERTION(rhs, "Null pointer passed to forget!");
```

```
4159 +          TBB_NS_ASSERTION(rhs, "Null pointer passed to forget!");
4160             *rhs = mRawPtr;
4161             mRawPtr = 0;
4162           }
4163 diff --git a/xpcom/glue/nsDebugTor.h b/xpcom/glue/nsDebugTor.h
4164 index 343e84e..55b6fc6 100644
4165 --- a/xpcom/glue/nsDebugTor.h
4166 +++ b/xpcom/glue/nsDebugTor.h
4167 @@ -15,7 +15,7 @@
4168  #endif
4169
4170  #include "nsXPCOM.h"
4171 -#include "mozilla/Assertions.h"
4172 +#include "mozilla/AssertionsTor.h"
4173  #include "mozilla/Likely.h"
4174
4175  #ifndef TOR_NASSERT
4176 @@ -349,7 +349,7 @@
4177    #define TBB_NS_CheckThreadSafe(owningThread, msg)
4178  #else
4179    #define TBB_NS_CheckThreadSafe(owningThread, msg)                \
4180 -    MOZ_ASSERT(owningThread == PR_GetCurrentThread(), msg)
4181 +    TBB_MOZ_ASSERT(owningThread == PR_GetCurrentThread(), msg)
4182  #endif
4183
4184  /* When compiling the XPCOM Glue on Windows, we pretend that it's going to
```

Listing 8: Sample Patch For Enabling Assertions In The JavaScript Engine

F Memory Allocator Replacement Patches

F.1 Replacement Sample

```
From da3f1399fcc9bbf8e0b66e9a3c649c58c0e46221 Mon Sep 17 00:00:00 2001
From: Tom Ritter <tom@ritter.vg>
Date: Wed, 21 May 2014 18:18:04 +0000
Subject: [PATCH] Sample Malloc-Replacing Library

---
 .mozconfig                          |    1 +
 memory/replace/moz.build            |    1 +
 memory/replace/realloc/Makefile.in  |   20 ++++++++++++++++++++
 memory/replace/realloc/moz.build    |   13 +++++++++++++
 memory/replace/realloc/realloc.c    |   32 ++++++++++++++++++++++++++++++++
 5 files changed, 67 insertions(+)
 create mode 100644 memory/replace/realloc/Makefile.in
 create mode 100644 memory/replace/realloc/moz.build
 create mode 100644 memory/replace/realloc/realloc.c

diff --git a/.mozconfig b/.mozconfig
index e9a9432..b957ebe 100755
--- a/.mozconfig
+++ b/.mozconfig
@@ -6,6 +6,7 @@ mk_add_options MOZ_MAKE_FLAGS="-j4"
 mk_add_options MOZILLA_OFFICIAL=1
 mk_add_options BUILD_OFFICIAL=1

+ac_add_options --enable-replace-malloc
 ac_add_options --enable-optimize
 #ac_add_options --disable-optimize
 ac_add_options --enable-official-branding
diff --git a/memory/replace/moz.build b/memory/replace/moz.build
index cb00e57..d378dce 100644
--- a/memory/replace/moz.build
+++ b/memory/replace/moz.build
@@ -7,3 +7,4 @@
 # Build jemalloc3 as a replace-malloc lib when building with mozjemalloc
 if not CONFIG['MOZ_JEMALLOC']:
     DIRS += ['jemalloc']
+DIRS += ['realloc']
diff --git a/memory/replace/realloc/Makefile.in b/memory/replace/realloc/Makefile.in
new file mode 100644
index 0000000..0893297
--- /dev/null
+++ b/memory/replace/realloc/Makefile.in
@@ -0,0 +1,20 @@
+# This Source Code Form is subject to the terms of the Mozilla Public
+# License, v. 2.0. If a copy of the MPL was not distributed with this
```

```
+# file, You can obtain one at http://mozilla.org/MPL/2.0/.
+
+DEPTH           = @DEPTH@
+topsrcdir       = @top_srcdir@
+srcdir          = @srcdir@
+VPATH           = @srcdir@
+
+include $(DEPTH)/config/autoconf.mk
+
+FORCE_SHARED_LIB = 1
+NO_DIST_INSTALL = 1
+
+VPATH += $(topsrcdir)/memory/build
+
+MOZ_GLUE_LDFLAGS = # Don't link against mozglue
+WRAP_LDFLAGS = # Never wrap malloc function calls with -Wl,--wrap
+
+include $(topsrcdir)/config/rules.mk
diff --git a/memory/replace/realloc/moz.build b/memory/replace/realloc/moz.build
new file mode 100644
index 0000000..7f48c22
--- /dev/null
+++ b/memory/replace/realloc/moz.build
@@ -0,0 +1,13 @@
+# -*- Mode: python; c-basic-offset: 4; indent-tabs-mode: nil; tab-width: 40 -*-
+# vim: set filetype=python:
+# This Source Code Form is subject to the terms of the Mozilla Public
+# License, v. 2.0. If a copy of the MPL was not distributed with this
+# file, You can obtain one at http://mozilla.org/MPL/2.0/.
+
+MODULE = 'memory'
+
+LIBRARY_NAME = 'replace_realloc'
+
+CSRCS += [
+    'realloc.c',
+    ]
diff --git a/memory/replace/realloc/realloc.c b/memory/replace/realloc/realloc.c
new file mode 100644
index 0000000..fd4e2b5
--- /dev/null
+++ b/memory/replace/realloc/realloc.c
@@ -0,0 +1,32 @@
+// This header will declare all the replacement functions, such that you don't need
+// to worry about exporting them with the right idiom (dllexport, visibility...)
+#include "replace_malloc.h"
+#include <stdlib.h>
+#include <stdio.h>
+
+static const malloc_table_t *funcs = NULL;
+static unsigned int total = 0, copies = 0;
```

```
+
+void replace_jemalloc_stats(jemalloc_stats_t *stats)
+{
+  printf("%d reallocs, %d copies\n", total, copies);
+  funcs->jemalloc_stats(stats);
+}
+
+void
+replace_init(const malloc_table_t *table)
+{
+  funcs = table;
+  printf("In init!\n");
+}
+
+void *replace_realloc(void *ptr, size_t size)
+{
+  void *newptr = funcs->realloc(ptr, size);
+  // Not thread-safe, but it's only an example.
+  total++;
+  // We don't want to count deallocations as copies.
+  if (newptr && newptr != ptr)
+    copies++;
+  return newptr;
+}
--
1.7.9.5
```

Listing 9: Sample Patch For Memory Allocator Replacement Library

F.2 CTMalloc Replacement Library

Note: This does not include the following files from http://src.chromium.org/blink/trunk/Source/wtf/. Some of these files were edited to prevent errors due to the use of undefined macros such as ENABLE.

- AddressSpaceRandomization.cpp
- AddressSpaceRandomization.h
- Assertions.h
- Atomics.h
- BitwiseOperations.h
- ByteSwap.h
- CPU.h
- Compiler.h
- Makefile.in
- PageAllocator.cpp
- PageAllocator.h
- PartitionAlloc.cpp
- PartitionAlloc.h
- ProcessID.h
- SpinLock.h
- WTFExport.h
- config.h

```
#include "replace_malloc.h"
#include <stdlib.h>
#include <stdio.h>

#include "config.h"
#include "wtf/PartitionAlloc.h"
#include <string.h>

static const malloc_table_t *funcs = NULL;
static unsigned int mallocs = 0, frees = 0, reallocs = 0, callocs = 0;

static PartitionAllocatorGeneric partition;
static bool initialized;

extern "C" {

void replace_init(const malloc_table_t *table)
{
  funcs = table;
  printf("In init!\n");
}

void replace_jemalloc_stats(jemalloc_stats_t *stats)
{
  printf("%d mallocs, %d frees, %d reallocs, %d callocs\n", mallocs, frees, reallocs,
       callocs);
}

void* replace_malloc(size_t size)
{
```

```
  mallocs++;
  if (UNLIKELY(!initialized)) {
    initialized = true;
    partition.init();
  }
  return partitionAllocGeneric(partition.root(), size);
}

void replace_free(void* ptr)
{
  //I believe this was a Chrome-only quirk. Going to attempt removing it
  //if (reinterpret_cast<uintptr_t>(ptr) >= 0x500000000000)
  //  return funcs->free(ptr);
  frees++;
  partitionFreeGeneric(partition.root(), ptr);
}

void* replace_realloc(void* ptr, size_t size)
{
  reallocs++;
  if (UNLIKELY(!initialized)) {
    initialized = true;
    partition.init();
  }
  if (UNLIKELY(!ptr)) {
    return partitionAllocGeneric(partition.root(), size);
  }
  //I believe this was a Chrome-only quirk. Going to attempt removing it
  //if (reinterpret_cast<uintptr_t>(ptr) >= 0x500000000000)
  //  return funcs->realloc(ptr, size);
  if (UNLIKELY(!size)) {
    partitionFreeGeneric(partition.root(), ptr);
    return 0;
  }
  return partitionReallocGeneric(partition.root(), ptr, size);
}

void* replace_calloc(size_t nmemb, size_t size)
{
  void* ret;
  size_t real_size = nmemb * size;
  if (UNLIKELY(!initialized)) {
    initialized = true;
    partition.init();
  }
  callocs++;
  RELEASE_ASSERT(!nmemb || real_size / nmemb == size);
  ret = partitionAllocGeneric(partition.root(), real_size);
  memset(ret, '\0', real_size);
  return ret;
}
```

```
81
82  void *replace_valloc(size_t size)
83  {
84    printf("AH!!!! valloc.\n");
85    return NULL;
86  }
87
88  void *replace_memalign(size_t alignment, size_t size)
89  {
90    size_t remainder = size % alignment;
91
92    return replace_malloc(size + remainder);
93  }
94
95  void *replace_aligned_alloc(size_t alignment, size_t size)
96  {
97    printf("AH!!! aligned_alloc\n");
98    return NULL;
99  }
100
101 int replace_posix_memalign(void **ptr, size_t alignment, size_t size)
102 {
103   size_t remainder = size % alignment;
104   *ptr = replace_malloc(size + remainder);
105   if(*ptr == NULL)
106     return -1;
107   return 0;
108 }
109
110 size_t replace_malloc_usable_size(usable_ptr_t ptr)
111 {
112   size_t s = partitionAllocGetSize(ptr);
113   return s;
114 }
115
116 size_t replace_malloc_good_size(size_t size)
117 {
118   return size;
119 }
120
121 void replace_jemalloc_purge_freed_pages()
122 {
123 }
124
125 void replace_jemalloc_free_dirty_pages()
126 {
127 }
128
129 }
```

Listing 10: Working Progress of ctmalloc replacement library

```
# -*- Mode: python; c-basic-offset: 4; indent-tabs-mode: nil; tab-width: 40 -*-
# vim: set filetype=python:
# This Source Code Form is subject to the terms of the Mozilla Public
# License, v. 2.0. If a copy of the MPL was not distributed with this
# file, You can obtain one at http://mozilla.org/MPL/2.0/.

MODULE = 'memory'

LIBRARY_NAME = 'replace_ctalloc'

CPP_SOURCES += [
    'malloc.cpp',
    'PartitionAlloc.cpp',
    'PageAllocator.cpp',
    'AddressSpaceRandomization.cpp',
    ]
```

Listing 11: Build File for ctmalloc library

G JavaScript Preference Options

The following code snippet indicates that when the browser is in "Safe Mode", several of these features are disabled regardless of preference. Safe Mode is determined if the environment variable `MOZ_SAFE_MODE_RESTART` is set, if the command line argument `-safe-mode` is supplied, or if the Shift or Option key is held on startup - more commonly it is entered when the user chooses to 'Restart with Add-Ons Disabled'.

```
static const char js_werror_option_str[] = JS_OPTIONS_DOT_STR "werror";
#ifdef JS_GC_ZEAL
static const char js_zeal_option_str[]      = JS_OPTIONS_DOT_STR "gczeal";
static const char js_zeal_frequency_str[]   = JS_OPTIONS_DOT_STR "gczeal.frequency";
#endif
static const char js_typeinfer_str[]        = JS_OPTIONS_DOT_STR "typeinference";
static const char js_pccounts_content_str[] = JS_OPTIONS_DOT_STR "pccounts.content";
static const char js_pccounts_chrome_str[]  = JS_OPTIONS_DOT_STR "pccounts.chrome";
static const char js_jit_hardening_str[]    = JS_OPTIONS_DOT_STR "jit_hardening";
static const char js_memlog_option_str[]    = JS_OPTIONS_DOT_STR "mem.log";
static const char js_memnotify_option_str[] = JS_OPTIONS_DOT_STR "mem.notify";
static const char js_disable_explicit_compartment_gc[] =
  JS_OPTIONS_DOT_STR "mem.disable_explicit_compartment_gc";
static const char js_asmjs_content_str[]    = JS_OPTIONS_DOT_STR "asmjs";
static const char js_baselinejit_content_str[] = JS_OPTIONS_DOT_STR "baselinejit.
    content";
static const char js_baselinejit_chrome_str[]  = JS_OPTIONS_DOT_STR "baselinejit.
    chrome";
static const char js_baselinejit_eager_str[]  = JS_OPTIONS_DOT_STR "baselinejit.
    unsafe_eager_compilation";
static const char js_ion_content_str[]        = JS_OPTIONS_DOT_STR "ion.content";
static const char js_ion_eager_str[]          = JS_OPTIONS_DOT_STR "ion.
    unsafe_eager_compilation";
static const char js_ion_parallel_compilation_str[] = JS_OPTIONS_DOT_STR "ion.
    parallel_compilation";

int
nsJSContext::JSOptionChangedCallback(const char *pref, void *data)
{
  //...
  bool usePCCounts = Preferences::GetBool(chromeWindow || !contentWindow ?
                                          js_pccounts_chrome_str :
                                          js_pccounts_content_str);
  bool useTypeInference = !chromeWindow && contentWindow &&
                  Preferences::GetBool(js_typeinfer_str);
  bool useHardening = Preferences::GetBool(js_jit_hardening_str);
  bool useBaselineJIT = Preferences::GetBool(chromeWindow || !contentWindow ?
                                             js_baselinejit_chrome_str :
                                             js_baselinejit_content_str);
  bool useBaselineJITEager = Preferences::GetBool(js_baselinejit_eager_str)
  bool useIon = Preferences::GetBool(js_ion_content_str);
  bool useIonEager = Preferences::GetBool(js_ion_eager_str);
```

```
  bool useAsmJS = Preferences::GetBool(js_asmjs_content_str);
  bool parallelIonCompilation=Preferences::GetBool(js_ion_parallel_compilation_str);
  nsCOMPtr<nsIXULRuntime> xr = do_GetService(XULRUNTIME_SERVICE_CONTRACTID);
  if (xr) {
    bool safeMode = false;
    xr->GetInSafeMode(&safeMode);
    if (safeMode) {
      usePCCounts = false; //javascript.options.pccounts.content or .chrome
      useTypeInference = false; //javascript.options.typeinference
      useHardening = false; //javascript.options.jit_hardening
      useBaselineJIT = false; //javascript.options.baselinejit.content or .chrome
      useBaselineJITEager = false; //javascript.options.baselinejit.
          unsafe_eager_compilation
      useIon = false; //javascript.options.ion.content
      useIonEager = false; //javascript.options.ion.unsafe_eager_compilation
      useAsmJS = false; //javascript.options.asmjs
    }
  }
```

Listing 12: dom/base/nsJSEnvironment.cpp

javascript.options.ion.content

This setting will disable Ion, the newer JIT engine. The main entry point for the Ion engine is a branch in `js::RunScript` in Interpreter.cpp. iSEC identified a number of bugs in the Ion JIT engine, as shown in section 3.1 on page 10.

javascript.options.baselinejit.content

This setting disables the Baseline Compiler.[56] Disabling this will also disable Ion:

```
static inline bool
IsIonEnabled(JSContext *cx)
{
    return cx->hasOption(JSOPTION_ION) &&
        cx->hasOption(JSOPTION_BASELINE) &&
        cx->typeInferenceEnabled();
}
```

Listing 13: js/src/jit/Ion.h

But if you disable Ion and leave this enabled, you will hit certain code paths that include parallel script execution (`js::ParallelDo`), a branch in the `js::RunScript` function, and a few other small areas. From what iSEC can tell, it does not make sense to leave this enabled if Ion is disabled.

[56] https://blog.mozilla.org/javascript/2013/04/05/the-baseline-compiler-has-landed/

javascript.options.typeinference

Note: The actual preference appears to be javascript.options.typeinference – and does not include a '.content' at the end.

As with the prior setting, disabling this setting will also disable Ion. But if you disable Ion and leave this enabled, it appears you will hit code paths in the JSScript, JSFunction, JSObject, TypeCompartment, and types classes, mostly contained in jsinfer.cpp.

iSEC search for bugs that may be related directly to Type Inference, and found several (799803, 822858, 785576, 781855, 811616, 820186, 807047, and 831055), implying that disabling this feature may in fact eliminate some exploitable code paths.