Break on Trusted Type violation (including report-only mode)

Attention: Externally visible, non-confidential
Authors: alcastano@google.com, sigurds@chromium.org
Status: Inception | Draft | Accepted | Done
Created: 2020-10-27 / **Last Updated**: 2020-11-19

One-page overview

Summary

For debugging and fixing trusted type violations, it is useful to inspect the JavaScript environment at the violation site. For this purpose, we are proposing a new DOM breakpoint category: *break on Trusted Type violation*. This breakpoint triggers whenever a specific trusted type violation occurs, independent of whether CSP is in report-only or enforced mode (note that an exception triggers in enforced mode).

Platforms

Desktop

Team

alcastano@google.com, sigurds@chromium.org

Tracking issue

https://crbug.com/1142804

Value proposition

This feature helps improve the debugging story for Trusted Types by making it easier for developers to figure out what went wrong and why by leveraging DevTools' JavaScript debugging features.

Code affected

DevTools front-end, DOM Debugger, Chrome DevTools Protocol, V8 PausedReasone nums and Trusted Types implementation in the back-end.

Signed off by

Name	Write (not) LGTM in this row
bmeurer@chromium.org	
vogelheim@chromium.org	
sigurds@chromium.org	LGTM
koto@google.com	
caseq@chromium.org	
yangguo@chromium.org	
...	

Core user stories

- As a developer, I want to break on the first Trusted Type violation that occurs in report-only mode
- As a developer, I want to break on the first Trusted Type violation that occurs before the exception is raised.
- As a developer, when a break occurs due to a Trusted Type violation, I want to know what type of violation triggered the error.

Preliminary Work

Providing accurate TT violation information

The current backend implementation of Trusted Types does not have a straightforward way of finding out whether a Trusted Type policy violation occurred, which is fundamental to know in order to break on Trusted Type violations. Such a limitation already led to some suboptimal workarounds in the code for detecting whether the violation was due to a duplication.

Therefore, in order to add support for breakpoints on Trusted Violations in a clean way, while also improving the organization of the existing code, it is required to refactor Trusted Type policy creation to provide an output enumerator providing accurate information about the exact violation that occurred, regardless of being report-only or enforced.

CL: https://chromium-review.googlesource.com/c/chromium/src/+/2517061

Design

The main goals of this design are to add support for breakpoints on Trusted Type violation in a way that it integrates seamlessly with the codebase, while making it easily extensible for future CSP violations that could be useful for setting breakpoints.

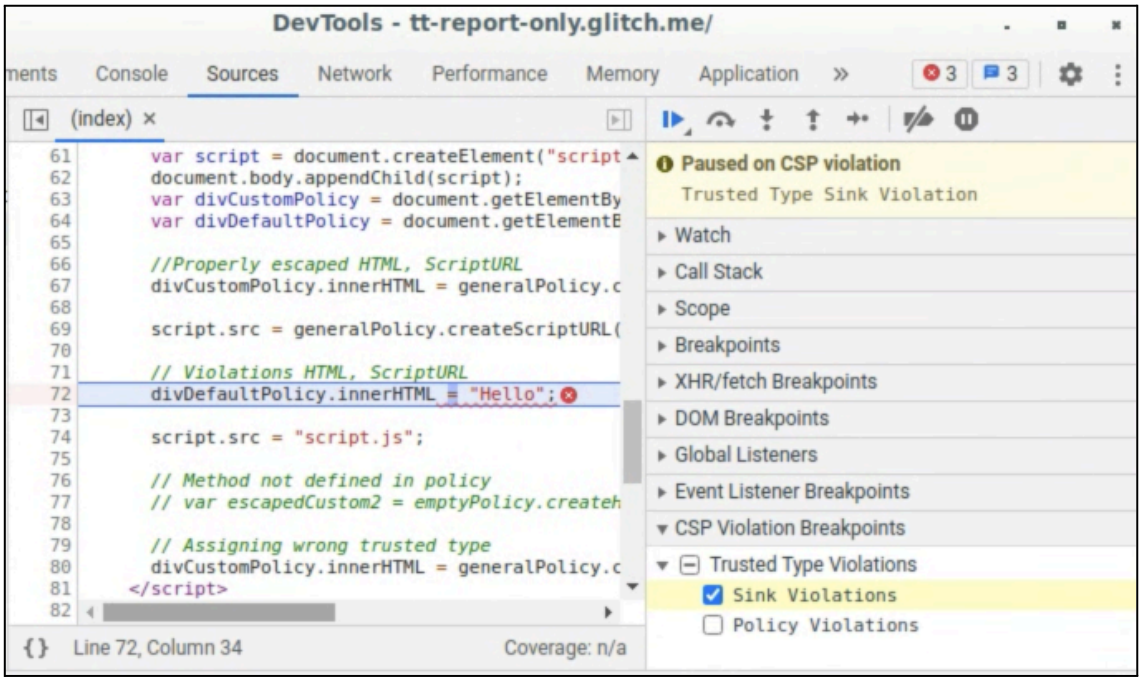
We are currently not considering more fine grained conditions such as break on report only violation as we don't have a good use-case.

Front-end design

The front-end needs to expose the function to set/unset this breakpoint in the UI. Currently, there are several sections for different kinds of issues like DOM Breakpoints or Even Listeners breakpoints. We introduce a new section in the right sidebar for CSP Violations. Currently the section will only contain Trusted Type violation but in the future it could be easily extended to contain breakpoints on other CSP or Security violations.

In particular, the UI follows the same approach as Event Listener having collapsible categories with breakpoint options inside. In this case, the category is Trusted Type violations and the options Sink and Policy violations.

Finally, alike the rest of breakpoint types, when a breakpoint is triggered the corresponding breakpoint item in the category is highlighted and contextual information is added in the right upper box of the panel.



For the implementation:

- Update CDP to expose DOMDebugger.setBreakOnCSPViolation and the required parameter types.

- Create the abstract classes CategorizedSidebarPane & CategorizedBreakpoint to group code that is common to EventListener and CSPViolations breakpoints, and other future sections which could use breakpoints grouped into categories.
- Introduce a new SidebarPane based on the new CategorizedSidebarPane. For that purpose (browser_debugger.js) and (module.json) should be extended.
- The SidebarPane will interact with DebuggerManager to send the setBreakpoint message to the backend.
- Introduce the list of CSP breakpoints in DebuggerManager.
- Update DebuggerPausedMessage to show a custom message when the Paused::Reason is CSPViolation.

CLs:

- Protocol update: 2520824
- CSP Pause reason (Devtools) 2520827
- Abstract class refactoring: 2529160
- Breakpoint on CSP violations (DevTools): 2517571

The back-end design

- Introduce Paused::Reason::CSPViolation in V8 to be able to distinguish in the frontend those events related to a break coming from a Trusted Type violation.

- Add a new CDP method DOMDebugger.setBreakOnCSPViolation() taking a subset of the strings ["TrustedTypeSinkViolation", "TrustedTypePolicyViolation"] as argument. Passing the empty list clears the breakpoints. Passing invalid arguments would also lead to clearing the breakpoints. The implementation of the method stores the passed flags in an appropriate way on the InspectorDOMDebuggerAgent object.

- Introduce a new probe (core_probes.pidl) with the name OnContentSecurityPolicyViolation(ExecutionContext*, ContentSecurityPolicyViolationType). Such a probe will be used in the functions TrustedTypeFail and TrustedTypePolicyFactory::createPolicy after we have determined that there was a violation (but independently of whether it was allowed, and before we throw the exception).

- Modify core_probes.json5 to indicate that the InspectorDomDebuggerAgent handles the probe.

- Add to InspectorDOMDebuggerAgent the member function OnContentSecurityPolicyViolation where we check whether the violation type is in the list of stored flags, and if so, we break the program.

- Introduce a new Paused::Reason::CSPViolation enum in V8 to use on breaks due to CSP violations.

CLs:

- CSP Pause reason (V8) 2519513
- Breakpoint support on Blink: 2517519

Rollout plan

Waterfall

Core principle considerations

Speed

This will introduce a core probe on the failure path of trusted types. Since this should only trigger in case of a failure, and the probe only does something if DevTools is attached, we don't expect a speed impact for Chrome users, and negligible impact for DevTools users.

Security

This will make it easier for Web Developers to improve security on their sites. No impact other than that is expected.

Simplicity

The implementation is in line complexity-wise with the implementation of similar features.

Accessibility

Standard UI elements will be used for this feature, so no impact is expected.

Testing plan

Inspector-protocol tests for the back-end interface, and e2e tests for the front-end.

Followup work

- This work can easily be extended for the other (i.e. non-trusted-type) CSP-violation features. However, we don't have a strong use-case for wanting to break on violations that are not related to Trusted Types, so we are holding off on this. Should we decide that we want this, the changes to the implementation should be minor (i.e. mainly moving the core probe to a better suited place).
- Allow to break only on report-only or non-report only TT violations. Can be useful if there are different headers or simply if the user only cares about one type of violation.
- Integrate with existing CSP breakpoint (from probe::ScriptExecutionBlockedByCSP)

Observations

- Since we reuse the same logic for CSP Violations as for Event listeners, we send one message to the backend per violation selected. That can be non optimal when you select a group of checkboxes at the same time, since you could just send a message with all the selected violations. However, since such a feature is only supported in CSP for the

moment we leave it as it since the negative implications are small while allowing better code reuse.