February 21, 2018

Isolation of networking process in a browser

Kinuko Toyama

Matthew Menke

Scott Graham

John Abd-El-Malek

Yuzhu Shen

See next page for additional authors

Inventor(s)
Kinuko Toyama, Matthew Menke, Scott Graham, John Abd-El-Malek, Yuzhu Shen, and Randall Smith

Isolation of networking process in a browser

ABSTRACT

Computer programs and applications are often designed such that sub-tasks of a program are executed by a separate process spawned by the main program or application. An advantage to such a design is resilience, e.g., even if one process fails, the remaining processes continue execution, even as the failed process is started anew. In the context of web browser applications, networking requests are typically not spawned as separate processes but rather handled by the main browser process itself. This can lead to problems of stability, security, and resilience. The techniques of this disclosure spawn the networking tasks of a web browser into a separate process, and can provide improved stability, security, and resilience.

KEYWORDS

● Browser security
● Process isolation
● Broker process
● Sandboxing

BACKGROUND

To enable a user to browse the internet, modern web browsers execute numerous tasks, e.g., fetching data over the network, rendering HTML pages in a manner appropriate to the platform, maintaining multiple tabs, etc. The task of fetching data, and ancillary tasks such as cookie management, cache maintenance, etc., are together referred to as networking tasks. Networking task is a critical task such that a failure in its execution can cause a failure in the browser process. Furthermore, a security breach of a networking task can compromise browser security, which can affect the privacy and security of user data.

One approach to improving the security and resilience of complex computer programs is to spawn the sub-tasks or services of a program into separate processes. A spawned task is scheduled by the operating system and runs in its own address space. Importantly, failure of any one process does not cascade into the failure of other tasks. Similarly, security breaches that occur within one process are contained therein.

Currently, browsers themselves spawn processes, e.g., each tab within a browser is often an independent operating system process. Isolating or sandboxing each tab has been shown to improve security and resilience. For example, if one tab fails, other tabs and the browser application continue to function.

DESCRIPTION

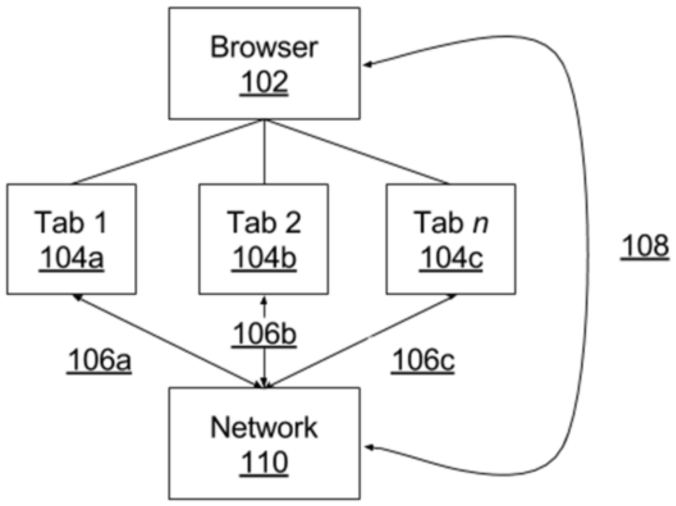

Fig. 1

To achieve improved security and resilience for a web browser, the techniques of this disclosure group networking-related tasks of a browser into one or more networking processes, that are distinct operating system processes. This is illustrated in Fig. 1, which shows a browser process (102) that spawns several child processes, e.g., tab processes (104a-c), and a network process (110).

Networking requests that originate from the browser application or any child process are routed through the network process. For example, the tab processes route their networking requests, e.g., fetching of data, through respective communication channels (106a-c). As a further example, the browser communicates, e.g., sends or receives configuration information, via a communication channel (108). To avoid latency through process hops, the browser and other child processes (referred to as consumer processes) make requests directly to the network process.

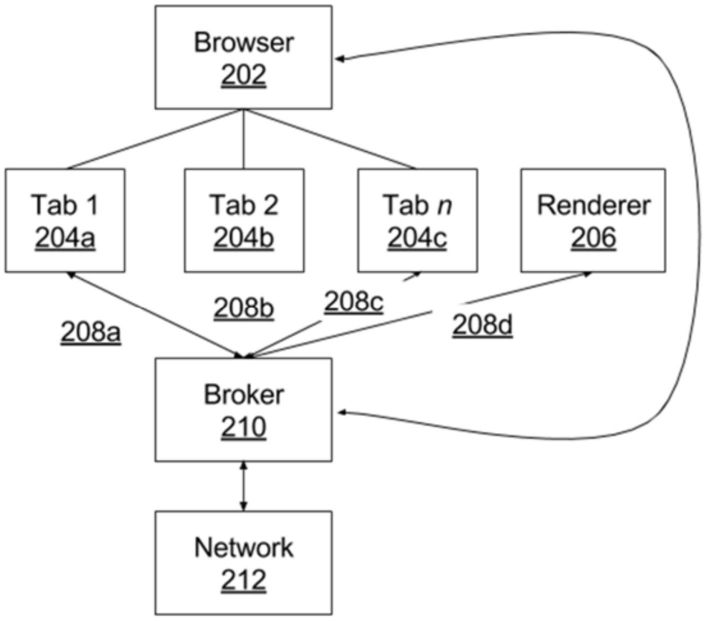

Fig. 2: A broker process that mediates communication between consumer processes and the network process

To further improve isolation, the browser process can spawn a central broker process that mediates communication between child processes of the browser and the networking process. This is illustrated in Fig. 2, wherein child processes of a browser (202), e.g., tab processes (204a-c), renderer processes (206), etc., communicate with network process (212) via a broker process (210) over respective communication channels (208a-d). In some instances, the broker process may be implemented as part of the browser process. The central broker monitors the network process, and if it detects a crash of the network process, the broker process restarts the network process. The communication links between the network process, broker process, and consumer processes are bi-directional. For example, the network process contacts the broker process whenever a user interface such as a username/password form, etc. is to be displayed.

Further, the network process can be optionally restricted, wherever the operating system provides such capability. For example, when a child process, e.g., a tab, is provided a connection to a network service, the connection can be restricted to certain URLs. Restricting the connection improves security. For example, in current browsers, a security breach may occur, e.g., if a browser tab connects to a malicious site that exploits bugs or loopholes in the tab process to run malicious code. By restricting resources that a process can gain access to, the malicious code is quarantined and thereby, browser security is improved. Other processes of the browser, e.g., renderer or other tab processes, continue to run legitimate code even if one tab process is compromised.

The network process is configurable to assign different priorities to ongoing requests. Further, the network process is configurable to simulate different network conditions, e.g., to test website performance on different networks. For example, a website developer may need to test website behavior and resilience when accessed via 3G wireless network. The network process can be configured to simulate a 3G wireless network for the purpose of test and development, e.g., by serving packets at a latency and packet-drop rate that is similar to a 3G wireless network.

By sandboxing or isolating the network process of a browser per techniques of this disclosure, one or more of the following benefits accrue.

- **Stability:** A crash of the network process does not lead to crash of the entire browser; rather, the network process can simply be restarted.
- **Security:** If the network process experiences a security breach, the potential harm is limited.
- **Flexibility:** The network process can be started independent of and without starting the rest of the browser. This is useful, e.g., to make networking requests when the browser isn't running.
- **Code simplification:** Network requests are made in the same manner regardless of the process that originates the request. This makes code development and maintenance simpler and more robust.

CONCLUSION

The techniques disclosed herein spawn the networking tasks of a browser into an independent operating system process, such that a failure in execution or security of the networking process is quarantined to the network process and doesn't spread to other parts of the browser. Per the techniques, a failure in network process execution allows the remaining parts of the browser to continue running, even as the crashed network process is resuscitated. In addition to a more secure and robust computing environment, the techniques enable greater flexibility, e.g., the network process can be run without browser invocation.