

Caja External Security Review

We went into the review knowing that Caja is not yet secure, and fully expecting our reviewers to find holes, which they did. The question was not "Is Caja secure today?" but rather "How secureable is Caja? Are we on a path that can take us to our goals? What should we change to be on a better trajectory?" We distinguished our short term and long term goals. Our short term goals involve deploying individual Caja gadgets in an iframe, or perhaps a set of non-precious gadgets within the same iframe, and running these iframes on a different origin than their containing page. Our long term goals involve inlining multiple mutually suspicious gadgets directly into the containing page, protecting the containing page from its gadgets, and protecting these gadgets from each other.

Regarding the short term goal, the review was generally positive. The outward facing Caja design seems sound, and no fundamental insecurities were found. A number of security holes were found, but all were local bugs that we are quickly fixing. Our reviewers generally felt that we could achieve the degree of confidence needed for our short term goals by continuing with such incremental engineering.

Our reviewers were clear, however, that such incremental engineering may not be adequate to achieve our long term goals. They identified problems in our implementation architecture, in our processes, our lack of documentation, and in the sheer complexity of the task we have taken on. We will need to address all of these before we can responsibly recommend that anyone use Caja alone to protect valuable components from each other.

Environment

We wrote a [testbed container](#) which hosted two Cajoled gadgets on the same page. The container had a piece of code the gadgets should be unable to execute, and was also supposed to prevent any involuntary communication between the two gadgets, whilst allowing voluntary communication. The container was also supposed to prevent any communication between the gadgets except through exported objects. Note that this container corresponds to the harder long-term goals.

The challenge we set the review team was to find ways to violate these constraints. The review team was also asked to comment on the implementation of Caja, and the concept itself.

Bugs

The review team did find one way to run unsanitized code directly, and so circumvent all protection. This was due to a simple oversight and was quickly fixed. They did not find any way for one gadget to interfere with the other gadget except in the form of "trojan horses" - that is, it was possible for one gadget to hand the other a piece of the DOM which could then be used to manipulate neighbouring DOM elements, apart from a minor oversight where the names of elements were not processed by the compiler.

One hidden communication channel was found which used regular expression handling as a vector.

None of the specific implementation issues that were found are proving to be particularly hard to fix.

The Caja team anticipates that more of these will be found as time goes by, but expects them to remain fixable.

Shortcomings

Caja is a large and complex project, and as a result the review team had a number of concerns about its security and reviewability. We list the main ones below, and the Caja team's comments on them.

Hard to review. No map that states invariants and points to where they are enforced., which hurts maintainability and security

We intend to refactor the compiler so that there are clearer boundaries between stages. We also intend to move to an AST that is closer to the semantics of Caja, rather than being a node-by-node Javascript representation, and only render this to Javascript at the last moment. This will allow more confidence that security properties are not distorted by subsequent processing.

Inadequate test/knowledge coverage of Browser quirks and corner cases

We will increase our test coverage to include a defined set of browsers. We will offer prizes for bugs, including browser quirks. We will use fuzzing to increase test coverage for browser quirks.

More complex than competing schemes in terms of spec & implementation

The Caja team agrees that our system is more complex, but this is the price we are paying for more functionality, flexibility, and legacy compatibility: there is no competing scheme that offers similar functionality, flexibility, or compatibility whilst making any security claims at all. So, we feel that Caja is necessarily more complex.

Nevertheless, we must work to simplify Caja as much as we can, and there is definitely room for simplification. We do need to provide correspondingly complete documentation. This is in hand, starting with a series of articles on writing Caja gadgets and containers, and hosting the Caja compiler.

Time to market is a concern

We are concerned about this, too, but we feel we are now ready to start using Caja in the wild within iframes. We do not want to suggest at this stage that Caja will prevent all attacks, but it will certainly raise the bar and demonstrably does prevent attacks that work against containers not using Caja. We are working towards live deployments right now.

Insufficient internal review/familiarity by team members w/ security-critical pieces of code

As mentioned above, the code base is large and complex. We intend to ensure more even coverage of the code by all team members by involving the whole team in the refactoring efforts, particularly of code they have not worked on before. We would also note that this is an open source project and so we hope that the refactoring mentioned above will reduce the bar for external participants.

Unclear how great a burden we are placing on container implementors

We feel it is clear to us, and that the burden is not great. However, we will be addressing this concern through documentation mentioned above.

Documentation of TCB is necessary for reviewability and confidence

Some members of the review team (Ka-Ping Yee and David-Sarah Hopwood) argued strongly that documentation of the internal design of the TCB was necessary in order for the TCB to be reviewable. Other members, in particular David Wagner, argued that it was sufficient to document its external interface. We agree with the latter position; that is, that TCB boundaries should be clear and that APIs should be well documented, as is standard practice at Google in any case.

Inner/outer hull defence inadequate

Due to the complexities of the browser environment, we knew we would make mistakes, so we constructed Caja with a "double hulled" confinement architecture. For many of the mistakes we might make, an attacker would need to compose an "outer hull breach" (obtaining a function value they should not be able to call) with an "inner hull breach" (tricking trusted code into calling this function value on their behalf) in order to have an exploitable breach. To probe the robustness of our inner hull, our testbed provided a simulated outer hull breach (the "Keystone Kop"). The inner hull mechanism is essentially ad hoc, not resting on any firm body of theory. Our reviewers found inner hull breaches with great rapidity. Though we are fixing these rapidly as well, we do not know how close we are to a solid inner hull.

Fortunately, inner hull weaknesses should only be exploitable following an outer hull breach. While some weaknesses were found in the outer hull mechanisms, including ways to violate the integrity of the trusted Caja runtime, few of these weaknesses were turned into exploitable breaches during the review. With more time, no doubt more of these could have been exploited. But altogether the outer hull seemed like a solid and securable design. In time, the double hulled architecture should provide some defense-in-depth.

Some reviewers felt that the double-hulled technique was insufficient, since the design does not rest on any sound theoretical basis and inner hull breaches were found relatively easily, so in effect we are relying on the outer hull only. These reviewers expressed skepticism that there was any reason for confidence that the double hull as currently designed would, over time, provide defense in depth (while allowing that alternative designs might do so).

Whilst we do agree to some extent with these comments, we feel that nevertheless

there is some value in the design - and it is not clear that inner hull breaches will continue to be found so easily as the system continues to be attacked, and, in any case, the outer hull was reasonably robust.

TCB too large, complex and interdependent

The TCB is structured in a way that does not facilitate separate review of its major components. A security reviewer must keep in mind too much of the design and implementation when reviewing any particular piece of it.

We agree. In particular some major components, like DOMita/DOMada, should be written in Caja instead of Javascript. We intend to refactor the TCB to be as small as possible.

Need documented process for announcing, revising, and distributing updates. Managing expectations of exploit reporters and container maintainer.

We agree. We intend to follow the pattern of existing open source security practice - working closely with exploit reporters and security researchers to produce prompt fixes to problems, a well documented process for managing vulnerability reports and an announce-only mailing list for updates, open to all.

The Review Team

The review was conducted by the following external reviewers:

Collin Jackson
David Wagner
David-Sarah Hopwood
Felix Lee
Ka-Ping Yee
Tyler Close

in conjunction with the Caja team:

Adrienne Felt
Ben Laurie
Eric Sachs
Ihab Awad
Jasvir Nagra
Mark Miller
Mike Samuel
Mike Stay

Edited by Ben Laurie, 19 July 2008.