**RTB Project**

# OpenRTB Security Advisory 15-01

Version 0.99.0.1

## ***Introduction***

*The RTB Project, formerly known as the OpenRTB Consortium, assembled in November 2010 to develop a new API specification for companies interested in an open protocol for the automated trading of digital media across a broader range of platforms, devices, and advertising solutions. This document is the culmination of those efforts and can be found at: www.iab.com*

## ***ABOUT THE IAB'S TECHNOLOGY LAB***

*The IAB Technology Laboratory is a nonprofit research and development consortium charged with producing and helping companies implement global industry technical standards and solutions. The goal of the Tech Lab is to reduce friction associated with the digital advertising and marketing supply chain while contributing to the safe growth of an industry.*

*The IAB Tech Lab spearheads the development of technical standards, creates and maintains a code library to assist in rapid, cost-effective implementation of IAB standards, and establishes a test platform for companies to evaluate the compatibility of their technology solutions with IAB standards, which for 18 years have been the foundation for interoperability and profitable growth in the digital advertising supply chain.*

***Further details about the IAB Technology Lab can be found at: http://www.iab.com/organizations/iab-tech-lab/****. The OpenRTB Work Group is a working group within the IAB Technology Lab.*

*This document can be found at www.iab.com*

*IAB Contact Information:*

*Melissa Gallo*
*Director of Product, Programmatic Automation and Data*
*IAB Technology Laboratory*
*melissa@iab.com*

## ***License***

## Table of Contents

## Change Log

| Version | Date | Section Link | Change |
|---|---|---|---|
| 1.0 | Nov 2015 | | Original Version |

# 1 Introduction

## 1.1 On Security Advisories

The IAB Tech Lab and its affiliates investigate reports of security concerns that may affect OpenRTB and related protocols, as well as implementations of these protocols. IAB releases advisories such as this one to support ongoing efforts to manage security risks and ensure a safe ecosystem for conducting programmatic advertising transactions. Advisories are intended to alert the community to noteworthy security issues, and do not necessarily designate exploitable vulnerabilities.

IAB recommends that implementors of OpenRTB review this advisory and determine whether the issue it describes affects systems that they operate or integrate with. Specific mitigation recommendations are discussed below.

OpenRTB Security Advisories include an identifier in the form YY-NN, where YY is the two-digit year number and NN is a zero-padded sequence number beginning with 01 for a given year. This is OpenRTB Security Advisory 15-01.

### 1.2 Limitations

This security advisory describes a means by which certain OpenRTB implementations can be abused by bad actors. OpenRTB itself is a data exchange protocol typically conducted between two servers. OpenRTB does not currently endorse any specific implementation code or business rules such as billing behavior, and can only issue guidelines on how best to implement the protocol. As such, OpenRTB cannot issue security patches, and can only issue guidelines on how to best implement the protocol.

### 1.3 Credits / Contributors

This document has been developed by the IAB Technology Lab's OpenRTB group. The OpenRTB Working Group mission and participation list can be reviewed at: http://www.iab.com/guidelines/real-time-bidding-rtb-project/

Contributors to this advisory included many members of OpenRTB working group, members of TAG and industry experts from security firms.

## 2 Advisory Details

### 2.1 Executive Summary

Certain implementations of real-time bidding can be abused by clients caching multiple ad impressions for an extended period, then replaying them in a rapid burst after a significant delay. This triggers a burst of win notifications and/or billing events each belonging to a unique auction_id. The delay and subsequent burst can cause frequency cap enforcement issues as well as disrupting the assumptions of many budget pacing algorithms. The result may be buying more more impressions per host and/or user than intended. Note that caching an impression is expected in some environments, such as with mobile apps and disconnected devices.

### 2.2 Issue Details

A malicious user can take advantage of some real-time bidding implementations to overcharge a buyer by generating multiple impressions in a way that can evade frequency caps.

In a normal ad impression, a client, such as a Web browser or mobile app, will request an ad. That ad may come from an exchange that conducts an auction, often via OpenRTB, to sell the impression. The auction is typically conducted by means of server-to-server communication between the exchange and multiple bidders. Each bidder will respond with a proposed price, ad markup to be served if the bid wins the auction, and other bid metadata as described in the OpenRTB specification. The exchange chooses a winning bid and passes the ad markup for the winning bid to the client that made the original request.

Normally the client will render that ad within a short period after the auction's close, unless the impression has been legitimately cached by, for example, the mobile advertising SDK. Once the ad has been rendered, it provides a form of win or render notification to the bidder. The bidder may use the render notification to enforce frequency caps or pacing restrictions.

A client intent on abuse can, instead of immediately rendering the ad, cache it indefinitely (illegitimately). It can then replay a batch of dated ads at a much later time, in rapid succession. This repeated delayed replay can cause bidders to record, and potentially pay for, invalid impressions. The impact of these replays can be magnified by impersonating or piggybacking onto high-value users. In addition, the replays may potentially also cause billing discrepancies between the bidder and exchanges.

This abuse is automated via a “botnet” and has been observed originating at hosts owned by organizations that the bidder may be accustomed to seeing a medium to high amount of impressions from. This particular botnet has been named “Xindi” and and this type of delayed event attack has been seen before by RTB security vendors.

This issue may affect implementations of business logic, pacing, and billing built around RTB transactions using any version of OpenRTB or any legacy or proprietary RTB protocol. The issue is not a direct vulnerability in the OpenRTB protocol, it is exposed via unsecured, or non-idempotent and/or incomplete validity checks on the handling of events associated to bidding via an any current RTB protocol.

Some implementations of OpenRTB use server-side win notification events and client side render events. Others implementations use client (device/browser) side win and render events. For the first configuration the replay attack creates a large delta between the auction and render timestamps, as the server-side win notifications are not delayed. In the second situation the replay attack creates a large delta between the auction and win notification.

In future versions of the OpenRTB protocol there may be updates and additional recommended best practices to future mitigate this particular issue. Additionally OpenRTB does not currently maintain or endorse any open source or proprietary implementation of the protocol and as such any patches to fix this issue must be made to each implementation found to have the vulnerability.

## 2.3 Suggested Remediation

IAB recommends that OpenRTB bidder implementations include the following three measures to protect against the abuse described in this advisory. In order to minimize discrepancy, IAB recommends that the bidder “penalty box” described in section 2.3.2 be implemented before, or concurrently with, the win notice timeout described in section 2.3.3.

### 2.3.1 Detection and Filtering

Exchanges and Bidders should implement impression scoring and filtering via internally developed technology and/or vendor solutions to identify and suppress selling or bidding on inventory that appears to be illegitimate or unnatural. The responsibilities on both sides are symmetrical.

*Responsibility of the exchange*

Make best effort to detect, classify and reject “non-human traffic” requests for ads to the exchange via the following best practices:

1. Build and/or license a traffic and event filtering mechanism
2. Filter impressions from known spiders via user-agent classification.

3. Filter impressions from suspected NHT via the detector and suppress the solicitation of bids for filtered events to all bidders (internal and external)
4. Process no-bid reason codes for bidders that support them and act on the aggregate signals to enhance filtering.

*Responsibility of the bidder*

Make best effort to detect, classify and reject “non-human traffic” requests for ads to the exchange via the following best practices:

1. Build and/or license a traffic and event filtering mechanism
2. no-bid on bid requests from known spiders via user-agent classification.
3. no-bid bid requests from suspected NHT via the detector.
4. Specify a no-bid reason code in either case.

### 2.3.2 Event Filtering and “Penalty Box”

To address this specific issue that is exposing a flaw in some bidders systems, bidders should monitor the rate of ads served or bid requests where the unique signature per unit time is excessive. By creating a unique signature of well chosen features, keeping a counter the bidder can place the signature in a “penalty box”. It is recommended that any signature’s event rate (bid request, bid, win, render, etc ) be examined for outlier status in real time as part of the penalty box.

This monitoring should be implemented at all budgetary levels as well as the campaign level. The threshold and signatures to be used should be chosen with data and business risk judgement. A simple example would be no more than 10 outbound bids per minute per deviceID or userID.

See the Stitelman 2013 paper in the References section for an approach to the penalty box concept.

### 2.3.3 Win Notice Timeout

Bidders should implement a timeout when the time difference between the win notification or render notification timestamp and the auction event timestamp exceeds some threshold value. Some examples values are below. The values used by any bidder should be chosen with data and use cases in mind.

- 60 seconds for desktop web browsers, mobile web browsers
- 10 minutes for mobile app banner or native ads that may be cached in a queue
- 10-20 minutes for mobile and video interstitials
- Very long or none for audio/server-side stitching/esoteric cases

For win or render notifications that are triggered after the (above) applied timeout multiple times at a rate that would be considered an outlier, the signature should be flagged and put in the “Penalty Box” as per section 2.3.1.

As a best practice any event that is rejected due to being outside the timeout period, the events should be logged and counted for any billing discrepancy conversations.

### 2.3.4 Price Encryption and Auction ID checks

Bidder should validate the auctionID as being authentic and may optionally use price encryption on the basis of encryption keys between the exchange and bidder to reduce the risk of win-notifications being manufactured and falsified.

### 2.3.5 No-Bid Reason Code

Bidders who have identified signs of abuse should respond to a bid request with no-bid reason code 4 indicating that they believe the traffic to be suspected non-human traffic. Doing so allows other participants in the ecosystem to use these signals to block suspect traffic earlier in the bidding process. See OpenRTB 2.3 nbr field and List 5.19.

# 3 Appendix

## 3.1 References

[1] [Stitelman 2013] Ori Stitelman, Claudia Perlich, Brian Dalessandro, Rod Hook, Troy Raeder, Foster J. Provost: Using co-visitation networks for detecting large scale online display advertising exchange fraud. KDD 2013: 1240-1248 http://chbrown.github.io/kdd-2013-usb/kdd/p1240.pdf

[2] http://media.pixalate.com/white-papers/xindi.pdf