Bidding and Auction Monitoring and debugging Plans (WICG)

Privacy Sandbox Protected Audience Team
- Itay Sharfi: itays@google.com,
- Akshay Pundle: apundle@google.com

Sep 2023

High level Summary

Bidding & Auction Services (B&A) Recap

- Two services for SSP to execute Protected Audiences auction in the cloud
- Two services for DSP to generate bids for Protected Audiences auction in the cloud
- Services are required to run in Trusted Execution Environment on a public cloud platform.
- Server side auctions supports all on-device auction features
- All bidding, scoring, reporting logic runs on high end servers
- Bidding & Auction Services can depend on BYOS Key/Value Services
- Scale to more Buyers, Sellers in auction
- Optimize latency & performance
- Confidentiality of Adtech's code and signals are protected with B&A
    - With a focus on preventing re-identification of user outside of TEE
- Seller can decide for each auction whether to run on-device or in B&

B&A Privacy and Security Recap

- Bidding and Auction services provide a secure, sandboxed environment on the cloud
  - Protects user data from unauthorized access
  - Protects Adtech data
- Bidding and Auction server code is Open Source, on GitHub
- Adtechs will deploy services to Trusted Execution Environment on a public cloud platform
- Coordinators (third-party entities) will generate and manage the cryptographic keys for encryption and decryption
  - Adtechs do not have access to decryption keys and will not be able to access raw, unencrypted user data
  - A user's device would encrypt data used in Protected Audience auction, a.k.a **encrypted Protected Audience data**
  - Only the B&A service can decrypt **encrypted Protected Audience data**

Debugging and monitoring Roadmap

Alpha
Jul ‘23

No prod traffic

Cloud metrics integration

Debug & Local mode

Beta
Nov ‘23

<1% prod traffic

Differential Privacy metrics (DP Noise can be turned off)

Aggregate error reporting

Developer consented debugging

Explainer updates

Scale Testing
Jun ‘24

Production ready

DP noise cannot be turned off

Monitoring

Monitoring overview

- Privacy safe telemetry from TEE servers
  - System metrics (e.g. CPU utilization) and server specific metrics (e.g. Num. requests)
  - On-Server Aggregation and Differentially Private noising
- This is different from other sources of Protected Audience metrics like event report, loss reports and PrivateAggregation reporting.
- Integration with Cloud native monitoring (e.g. CloudMonitoring, Cloudwatch) through OpenTelemetry allowing:
  - Dashboards
  - Alerts
  - Analysis and other natively supported functionality

Reference: Monitoring Protected API Services

Monitoring overview

- Servers pre-define the set of metrics that can be collected for telemetry
- Metrics will be classified according to their Privacy impact.
    - Privacy sensitive metrics are aggregated and have noise added to preserve privacy through differential privacy mechanisms.
    - Metrics not impacting privacy have no noise added. They are aggregated for performance reasons before being published.
- Data is exported using Open Telemetry to cloud monitoring systems.
- Exported data can be used for dashboards, alerting and other natively supported functions in the cloud monitoring systems.

Privacy Safe Telemetry

- We use Differential privacy to add noise to privacy impacting metrics before they leave the TEE.
- The ad tech will be able to configure the subset of metrics they want to track from a published list of available metrics.
  - This config will be per service.
  - All servers of the same service will track the same metrics.
- A total privacy budget will be set cumulatively for all privacy impacting metrics tracked by the ad tech.
  - This budget will be shared among all that the ad tech has chosen to track.
  - Non Privacy-impacting metrics will not consume any budget and will not have noise added.

Debugging

Supported types of debugging

- **AdTech consented debugging**: Provides a way for AdTechs to gain access to debugging data and requests that can be replayed on local systems
- **Local, Debug mode**: Provides a way for AdTechs to use standard debugging tools.
  - This can be used along with AdTech consented debugging to replay and debug requests.
- **Aggregate error reporting**: Provides aggregate error counts for tracking errors in production systems.

Reference: [Debugging Protected API Services]

Consented debugging

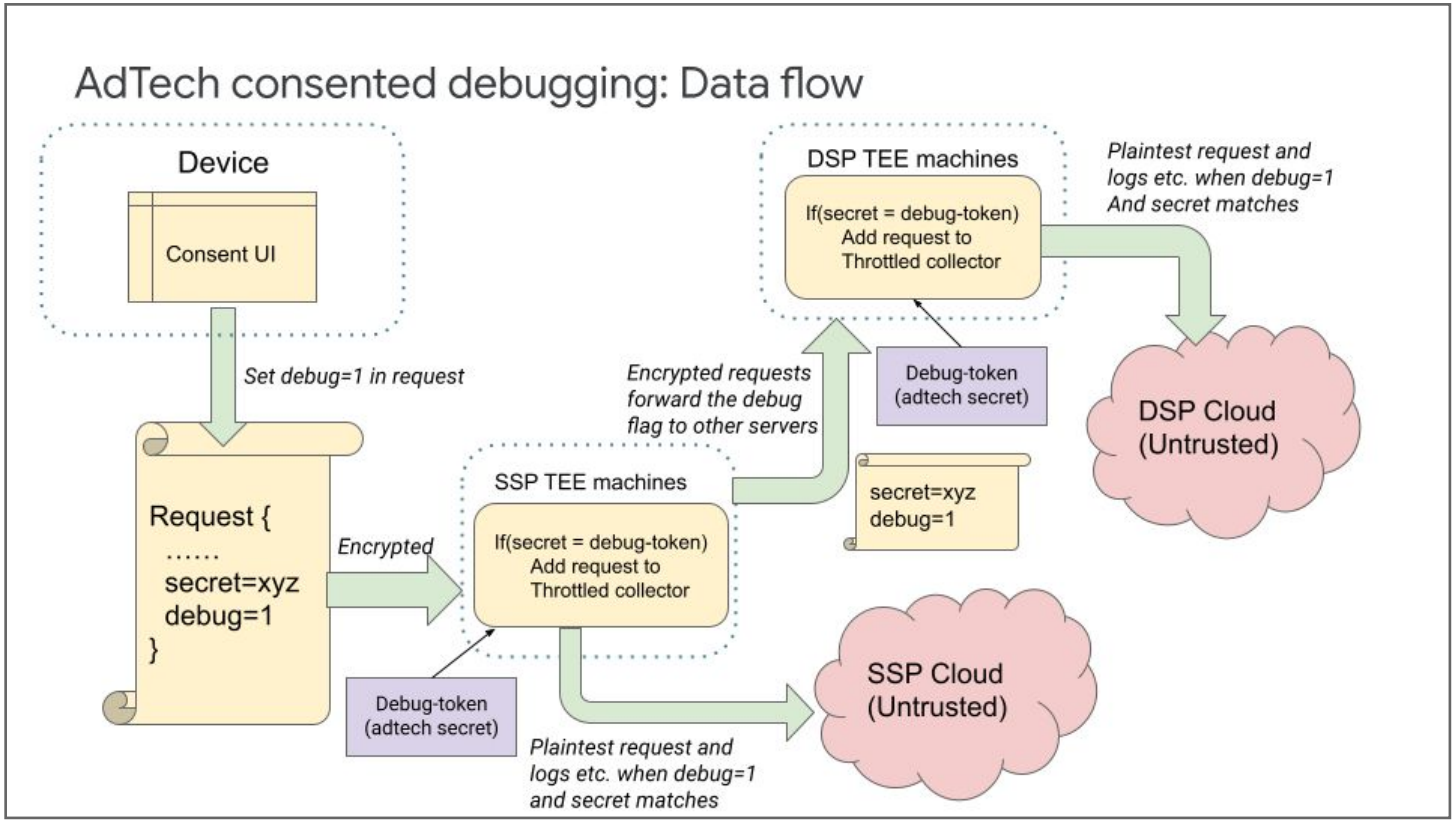
AdTech consented debugging

- AdTech consented debugging enables AdTechs to get plaintext requests and other debugging data from their servers.
  - The Protected Audience API normally protects this data and is accessible only to the end user (i.e owner of the data).
  - This feature is aimed at AdTech developers looking to debug the system, and not regular users of the system or API.
- The main use case and flow
  - AdTech wants to debug issues
  - They (the AdTech developer) act as the user, and grant consent on their client by initiating a special mode in Chrome or Android
  - This makes their request and associated data available for debugging on their system.

AdTech consented debugging

- A user (in this case, the AdTech developer) is fully in control of their Protected Audience API data, and chooses to make it available for debugging
- Debug reporting will not be provided for requests from any users besides those who have enabled this special mode.
- This consent allows the system to collect debugging data and provide it to the AdTech including:
  - The request in plaintext
  - Unnoised metrics
  - Logs (including console logs from JS execution)
- The consent works only for the servers of that ad tech. This happens through the use of a debug-token that is matched against the token provided by each server.

How it works



Local debugging and debug mode

Local debugging

- The Protected Audience API services will be runnable on the local machine
  - This means it will be possible to set up a local instance (e.g. B&A, K/V services) of a Protected Audience API server, send plaintext requests collected using Adtech consented debugging, and use standard debugging tools (since this will not be running inside a TEE)
  - This includes examining system logs, stack traces etc.
- Such a locally running server will not be able to decrypt or respond to production traffic

Debug mode

- TEE servers provide a special execution mode for debugging called Debug mode.
  - In this mode, the server runs inside the TEE, but cannot attest with the production key management systems, so they cannot decrypt any production traffic.
  - This mode provides some debug capability like console logs.
- Debug mode is useful for testing a more realistically deployed system compared to local testing.
  - The method of replay requests can also be used with servers launched in the debug mode.

Aggregate error reporting

Overview

- We will provide cumulative error counts per server
  - These will include request level and system errors
  - We will initially not support errors directly originating from AdTech provided javascript.
- Aggregate errors will also be published using Open Telemetry as mentioned in Monitoring Protected Audience API Services
- Error counts will be noised
- For consented debug requests, the true error counts will be reported without noise

Reference: Debugging Protected API Services

Feedback

Areas we are looking for feedback

- Coverage of use-cases
  - Is there sufficient coverage of use-cases? What are the gaps?
- Monitoring
  - What other metrics would you like to track?
  - What metrics would you track from JS (Custom ad tech defined metrics)
- Debugging
  - What are the use-cases that are not covered?
  - Would Breakglass access help? How and for what use-cases?
- Integrations / choice of technologies
- Complexity of adoption

References

- Monitoring Protected API Services
- Debugging Protected API Services