**VirusTotal Design Project**

This “whiteboard” project will be building a “VirusTotal” clone. This is a hypothetical design project that requires you to consider real world distributed systems challenges based on a publicly available product. You can read up more about VirusTotal by visiting their website at <https://www.virustotal.com> and going to **About Us > How it works**. You are also encouraged to try the system out by uploading a file so that you can get a feel for the end-user experience.

**Functionality**

Your task is not to replicate all of VirusTotal, but to understand the core functionality of VirusTotal and propose an architecture that will accomplish the following goals:

* Allow users to upload files to be scanned by multiple AV engines and metadata extraction scripts.
  + Engines and scripts may perform a wide range of operations such as: run a proprietary virus scanner, extract metadata from file headers, make calls to external service endpoints, execute uploaded files to observe their behavior, etc.
* Store metadata and AV engine results about uploaded files.
* Retrieve metadata about uploaded files (including file attributes, metadata and AV engine results).
  + Here is an example of metadata collected from file uploads: https://www.virustotal.com/en/file/0bf01094f5c699046d8228a8f5d5754ea454e5e58b4e6151fef37f32c83f6497/analysis/
* Ensure all major functionality is available via both a user interface and a public-facing APIs that users can build apps on top of.

**Requirements and Constraints**

When designing a system to support the aforementioned functionality, please also consider the following requirements and constraints:

* The system must support millions of users per day.
* Uploaded files can range in size from 100K to 1GB.
* Uploaded files should be retained as long as possible (ideally forever).
* The system must provide results as close to real-time as possible.
* Metadata and scanning services can run on a mixture of Linux or Windows (not all scanning services support the same OS).
* Your design should be able to accommodate the addition of new AV engines or metadata scripts. These additions should be as minimally invasive to the running system as possible (i.e. minimal or zero system downtime, no “big bang” deployments, etc.).

**What are we looking for?**

We’d like you to be as detailed as possible without needing to write any actual code. You should be prepared to talk about:

* The services that need to be built and their APIs
* The flow of data through the system and how we’ll process files efficiently
* Data organization including structure, schemas, sharding and partitioning
* Metrics and operational information (to ensure the system is running properly)
* Availability, scalability and fault tolerance
* Choice of underlying technologies
  + For the purposes of this assignment, we recommended choosing technologies that you're familiar with and are comfortable discussing

**Some sample questions you can expect might include:**

* How does the design address the basic requirements?
* How and where will we store the data?
* What is the structure and schema (if applicable) of our stored data?
* What technologies will we use?
* How will we handle failures in the system?
* What might a sample API request/response look like?
* How might we extend this architecture to add new features?
* Etc.

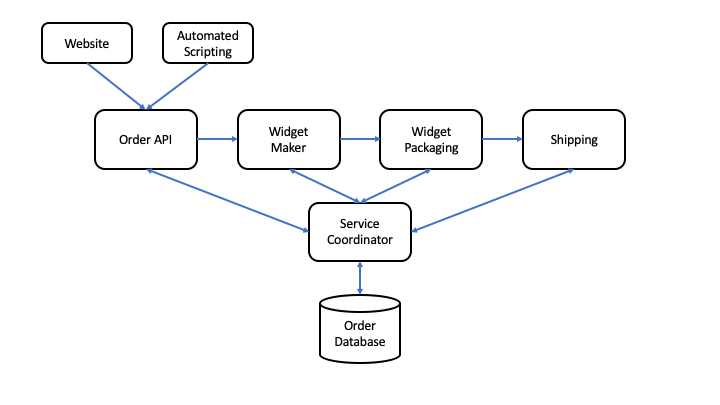
We will be validating that you have sound ideas, that you can collaborate with us, and that you can you translate what’s in your head into pictures that communicate your intentions effectively.

If interviewing in person, be prepared to collaborate with your interviewers on a whiteboard. If interviewing remotely, please prepare materials, such as a high-level block diagrams, to share with your interviewers in order to facilitate the discussion.

Interviews are meant to be bidirectional, so please also consider your own experience and whether or not you enjoy collaborating with us during the interview as well. This exercise is meant to be indicative of how we design new features and systems inside CrowdStrike.

If you have any questions at all, please feel free to ask!

**Sample Architectural Diagram (Simplified)**

****