**Any guidance on protecting legacy APIs that are not inherently designed for security? Without complete rewrite / re-engineering?**

The API Gateway pattern allows you to place security in front of the legacy (e.g. Plain Old XML / POX) APIs. You can also use the API Gateway pattern to create a virtualized REST API in front of the legacy XML API.

**As an API provider, how can I ensure credentials/tokens are not compromised by the client?** For e.g., I can force them to use a client secret, but they might have the secret hardcoded/insecure place that will allow malacious user access to it & compromise

good question - e.g. with many API providers they make it clear that the client must ensure that the client API keys are not compromised. So, it the responsibility is on the client. Often, an API Gateway is used at the client side to protect the keys. Many organizations use an API Gateway like the Axway API Gateway at the client side for this reason (security of API keys in the outbound direction)

**Which version oauth 1.0 or 2.0?**

For use cases that need end to end protection, I prefer signed security tokens which is older version of OAuth

**Recommendations for migrating from a username/password for an API to SAML or OAuth?**

I think either can work, see previous answer, if you need more specifics let me know

**What authentication mechanism would you recommend for api that are consumed by only internal applications? Should it still use api manager?**

In my view yes. For one thing its an admin convenience. For another its an API for security services that you would otherwise have to write yourself

**What about HMAC that can be used without TLS?**

HMAC provides digests over messages, usually as part of authentication and ensuring message integrity (detecting if it has been tampered it). it doesn't provide encryption of the data itself. So you need TLS for the encryption part. In general it is recommended TLS 1.2 is used, as a given, even when using HMAC for digest of the message itself.

**How can we leverage API gateways for context based authentication? Can we dynamically challenge for a second factor if detected that the context may have been compromised. In other words is there and authn/authz authorities that the API gateway can leveraged?**

Super cool topic, risk based auth is usually a mix of fingerprinting - transactions, usage, patterns. Very powerful, but requires some back end data stores and analytics as well.

**What's the difference between using SAML vs. OAuth from a security perspective?**

SAML arguably has a stronger security profile due to digital signatures, however OAuth has wide adoption because developers find it easier to use (and they do not have to deal with XML). Important - BOTH need TLS

**Where does OpenConnect Id fit into this picture?**

OpenID Connect (OIDC) provides a very userful "UserInfo" API endpoint to look up user attributes based on an OAUth Access Token. So it builds upon OAuth in a very useful way. At Axway we see it being used in this way, and is a useful part of ABAC (to obtain the attibutes used in ABAC)

**How can we take care of DDOS?**

DDOS is still taken care of at the network layer - e.g. using Akamai or AWS CloudFront. API Security doesn't replace protection for large scale DDOS

**Why is the statement - Oauth better than TLS? I was under the impression that both are needed in order to secure?**

Both are needed. OAuth is at the app level, TLS is at the network level. TLS protects the channel through which OAuth passes

**what is difference between oauth and hmac message?**

HMAC is for computing digests of messages, for example with Amazon AWS usage of two API Keys (Secret Key ID which is used for computing the digest, and the Access Key ID which is used for identification). OAuth is used for delegated authorization.

**What would be some security best practices for Hackathons?**

Great questions. I have seen the anti-pattern of hackathon organizers removing authentication from their APIs in order to "make it easier for them to be used in a hackathon". I don't recommend that. It is better to make use of an API Portal for the hackathon, where developers can authenticate and see sample API calls, including (for example) OAuth Access Tokens. At Axway our API Portal provides this generation of tokens, to help developers understand the security model of the APIs. A dedicated API Developer Portal for the hackathon also provides a simple "menu" of the APIs which can be used in the hackathon, with documentation, samples, etc.

**Is it assumed that the API Gateway and the APIs themselves are local to each other -- IOW, a private connection between the API Gateway and the API servers?**

In my view, its best if they are physically and logically separate, but logical separation alone is way better than nothing

**What about JWT? How does that fit in?**

JWT (JSON Web Tokens) are often used as a way to package attributes as JSON, and associate them with an OAuth Access Token. In this way, they provide a great way to pass attributes about the user to the API itself, so that fine-grained Authorization can be performed right at the API (or, the API usage can be personalized for the end-user based on the attributes in the JWT token)

**Building a private REST API that will be used only by web apps and mobile apps developed within our org. Evaluating both SAML and OAUTH, what would you recommend?**

Either can work. As a security person I feel like I am in a defensible position with either. I would recommend looking at sample code and building out a reference implementation to see which works with your dev team. Salesforce.com has some sample code to look at for example

**If there are older proprietary or domain specific communication protocols that lack security how to add security to these communications? E.g. medical software protocols.**

First step is usually to ensure you do NOT do lowest common denominator - work toward highest level for each hop. For true legacy, sometimes out of band can work. Try to initiate or validate out of band if possible

**What is the best way to go about helping developers learn how to prevent SQL injections?**

OWASP has some great information, including example mod\_security rulesets which protect against attacks like SQL Injection at www.owasp.org [at Axway we embed mod\_security in our API Gateway to enforce these rules]

API Security Webinar

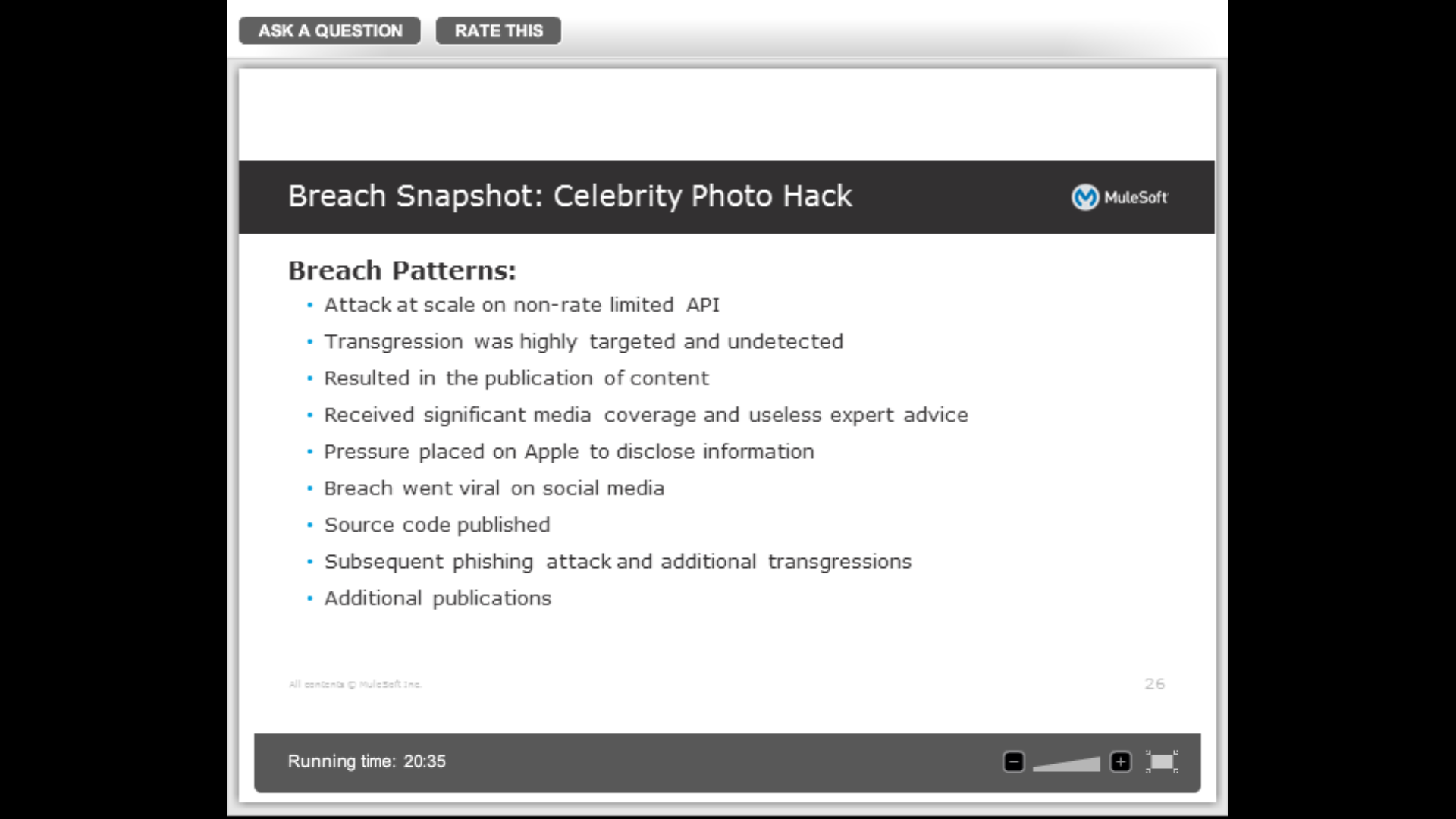
December 9, 2015

Common Breach Patterns

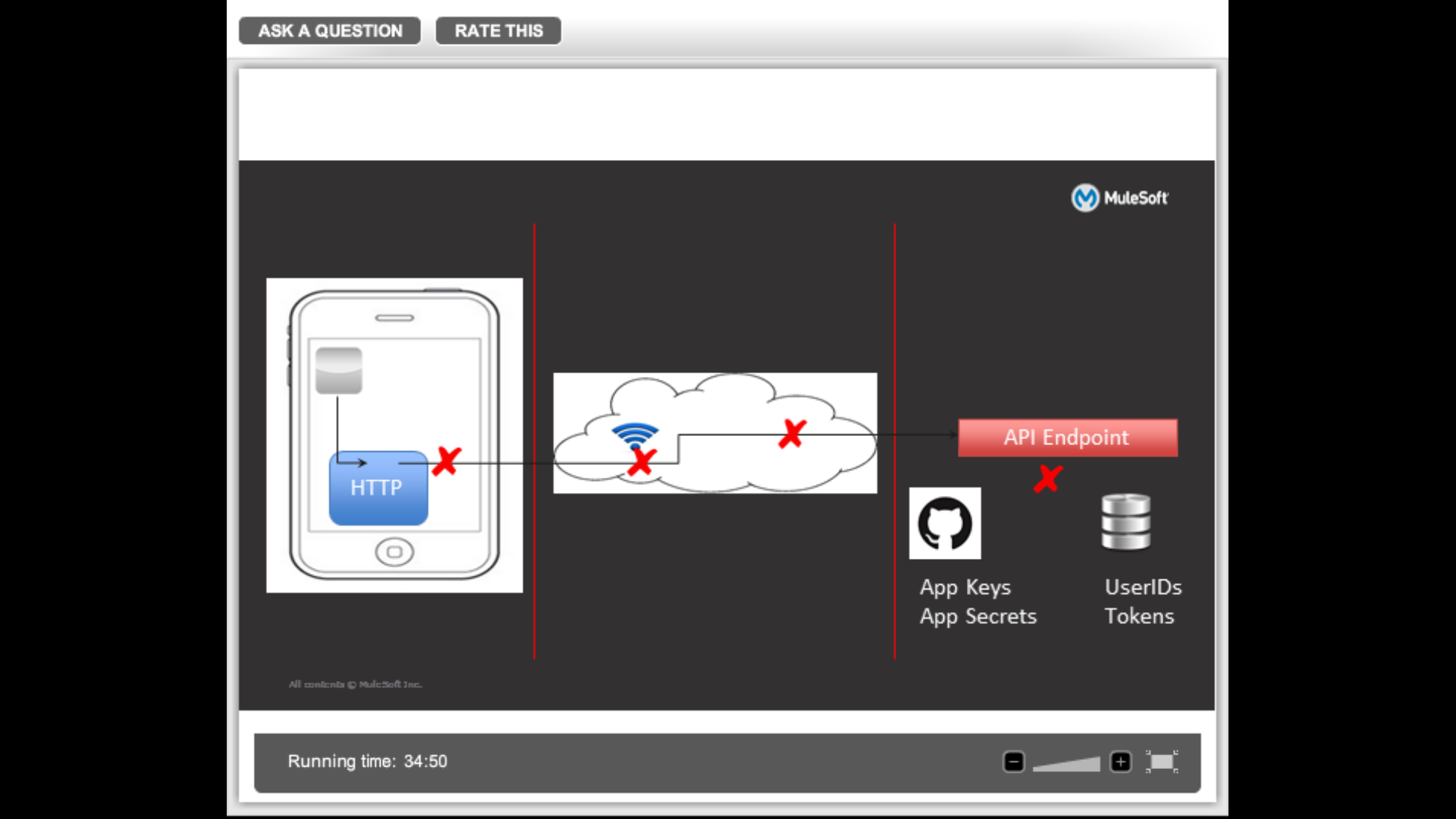
* Hackers seek potential for scale – APIs are sitting ducks
* Original transgression often targeted and undetected
* Leverages trusted Relationships (the downside of social nets!)
* Publication, sale or blackmail of content (include app secrets, & API Keys)
* Publication of source code
* Media coverage , useless expert advise
* Official company disclosure (sometimes)
* News goes viral on social media (useally –ve)
* Partners get sucked in
* Phishing attack, invariably malware
* Additional transgression
* Additional publication
* Mobile APIs are easy to reverse engineer

Apple Photo Hack

* Through mobile API
* API used only username/password
* Did not have any rate limiting on it – allowing brute force attack



* Mitigation Best Practices
  + Rate limiting
  + Secure communication b/w mobile apps and servers hosting underlying data
    - Assume open wifi and traffic snooping
  + Prescriptive guidance on credentialing (e.g. passwords)
  + MFA on code repositories
  + Encrypt OAuth credentials at rest
  + Do not store credentials in source code in plain text
  + Improve developer sensitivity to phishing attacks



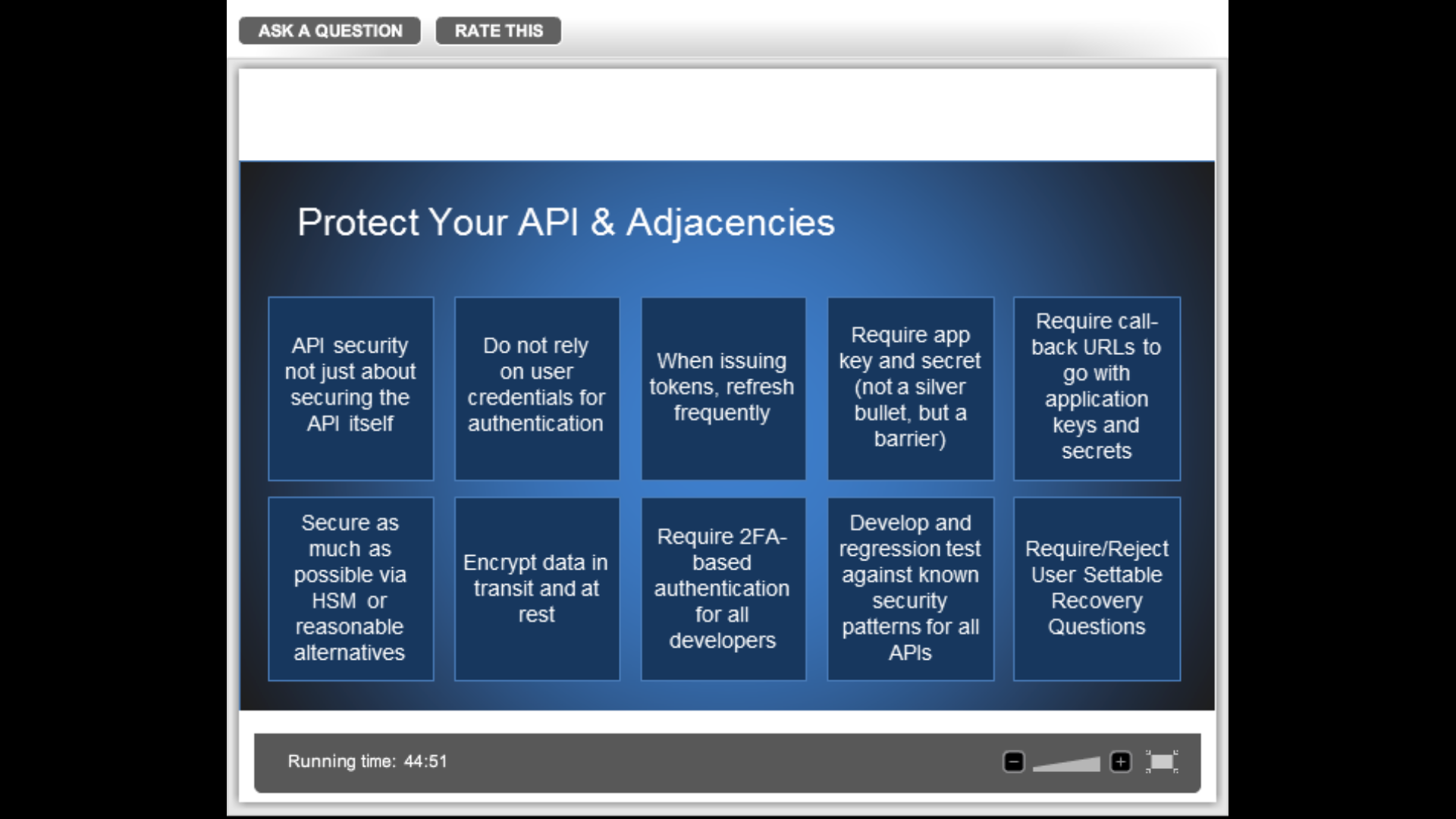
* There are apps available in Google Playstore that if installed on mobile, can be used to snoop traffic and show it to the user, even if the mobile uses https

Mitigating the mobile threat

* Threat is largely unresolved
* Certificate pinning is a stop-gap solution
  + End application is to the target application – it has to be a 2 way SSL handshake
  + DNS needs to be working
* Ongoing work at IETF towards better securing OAuth workflow
  + Proof of possession (service knows that the consuming application who is asking for information on behalf of the user is the consuming application who should be asking it)
* Canned solutions significantly lagging the bleeding edge

Challenges in API Security

* Massive proliferation of APIs where security was after-thought or non-thought
* User Id/Password Absurdity:
* Non-uniform implementation of App secrets and callback URLs
* Good security is expensive. Even the largest companies like google, etc. are not able to manage it
  + Talent
  + Resources like Hardware security module
* Administrative tools for key/OAuth management limited
  + Analytics
  + Revocation/reissue
* Unknown possibilities for 2FA with APIs
* IoT
* Documentation, disclosure, collaboration



* What are some of the first steps for organizations to start taking?
  + Take a Security first approach (security as a fore-thought)
  + Not storing OAuth in rest
* How much is human error part of the security breaches? How can orgs address it?
* As an API developer, how can
  + Every API provider should publish document which clearly document security best practices for developers to follow
* What predictions?

