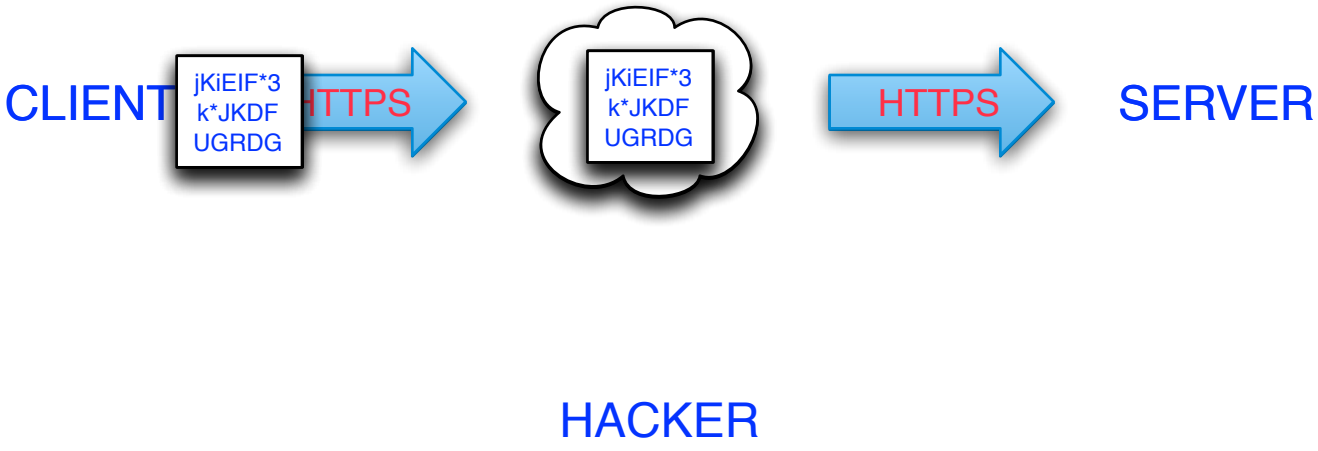
apigee

January 2014

Create and Manage APIs

Other Security Considerations

HTTP versus HTTPS



- SSL and TLS are the protocols that accomplish this
- The term "SSL" is used to refer to both SSL and TLS

What information is encrypted via SSL?

- Connection is made across network using IP address
  - IP address can generally be used to get a domain name
- Once connection is established, all data is encrypted
  - Including URL, headers, query parameters, verb and payload
- Snoopers know the destination server and how much data you are sending, but that is it

So then I can send passwords via query params?

```
GET https://example.com/v1/api/login?user=bob&pw=opensesame
```

- Passwords would be completely protected during the SSL transfer
  - But DON'T!
- The issue is data at rest – not data in motion
  - URLs are often logged in clear text server logs, including query parameters
  - GET requests are bookmarkable and visible in browser history
- URLs are generally not treated in a secure fashion, so don't use them to send sensitive data
  - Payload or headers are better

One-Way versus Two-Way SSL

- One-way SSL (server validation)
  - Server presents certificate, client does not
  - Client optionally validates the server certificate
  - Server must validate client via other means (HTTP message traffic)
    - Basic Auth, OAuth, etc.
  - This is standard web https

- Two-way SSL (mutual authentication)
  - Both client and server present a certificate
  - Client and server each validate the other's certificate

- One-way SSL is much more commonly used
  - Common to use two-way SSL for machine-to-machine connections, including Apigee to backend target

Apigee Keystores and Truststores

- Keystore
  - Used to store certificate(s) to be presented to remote server during SSL communication
  - Also stores private key used to encrypt SSL traffic to the remote server
- Truststore
  - Used to store certificates to compare with remote certificates received during SSL communications
  - Communication may be set to only allow communication with trusted servers
- Keystores and truststores are used for both client communication (via virtual hosts) and target communication (via target endpoints and target servers)

HTTP Persistent Connections

- HTTP 1.0 connections are not persistent
  - Use `Connection: Keep-Alive` header
- HTTP 1.1 connections are persistent by default
  - Client or server can send `Connection: close` header to tear down a connection
- Connection establishment and teardown are relatively expensive

- For SSL, we want to use a persistent connection if more traffic is likely to come from the client
  - For Apigee to backend, we almost always want a persistent connection
    - Traffic from all clients generally flow to the same few targets, so connection is likely to be reused quickly
  - For client to Apigee, balance the cost of caching connections with the likelihood of reusing connections

Virtual Host SSL Configuration

- Used to differentiate incoming traffic
  - Configured on Apigee Edge
  - Only for client requests, not target communication
  - For cloud, virtual hosts can only be configured by Apigee Support

```
GET https://api.enterprise.apigee.com/v1/o/{org}/e/{env}/virtualhosts/secure
{
  "hostAliases" : [ "myorg-prod.apigee.net" ],
  "interfaces" : [],
  "name" : "secure",
  "port" : "443",
  "sSLInfo" : {
    "ciphers" : [],
    "clientAuthEnabled" : true,
    "enabled" : true,
    "ignoreValidationErrors" : false,

    "keyAlias" : "myKey",
    "keyStore" : "myKeystore",

    "protocols" : [],
    "trustStore" : "myTruststore"
  }
}
```

Securing Calls to the Backend

- Generally the backend is locked down to only allow calls from Edge
  - Don't want apps to be able to call directly to backend

- Options for securing the communication to the backend
  - Credentials
  - OAuth (adds significant complexity to backend calls)
  - IP Whitelisting (can be spoofed)
  - Two-way SSL

Securing Backend Communication with Two-Way SSL

1. Obtain/generate client certificate for Edge

2. Create and populate a keystore on Edge containing Edge's cert and private key

3. Create and populate a truststore on Edge containing trusted certs

4. Configure the TargetEndpoint or TargetServer

- http://snap.apigee.com/ApigeeBackendMutualSSL

Data Masking

- Apigee Edge's Trace Tool allows developers to capture runtime traffic
- Some of the data exposed by the trace tool may be sensitive information, such as passwords, credit card numbers, or personal health information
- To filter this data out of the captured trace information, Apigee provides **Data Masking**
- Data masking can block values in XML payloads, JSON payloads, and variables
- Data masking configurations can be set globally for an organization

```
POST /v1/o/{org}/maskconfigs
```

or on specific apis

```
POST /v1/o/{org}/apis/{api}/maskconfigs
```

Data Masking

- Data masking payload

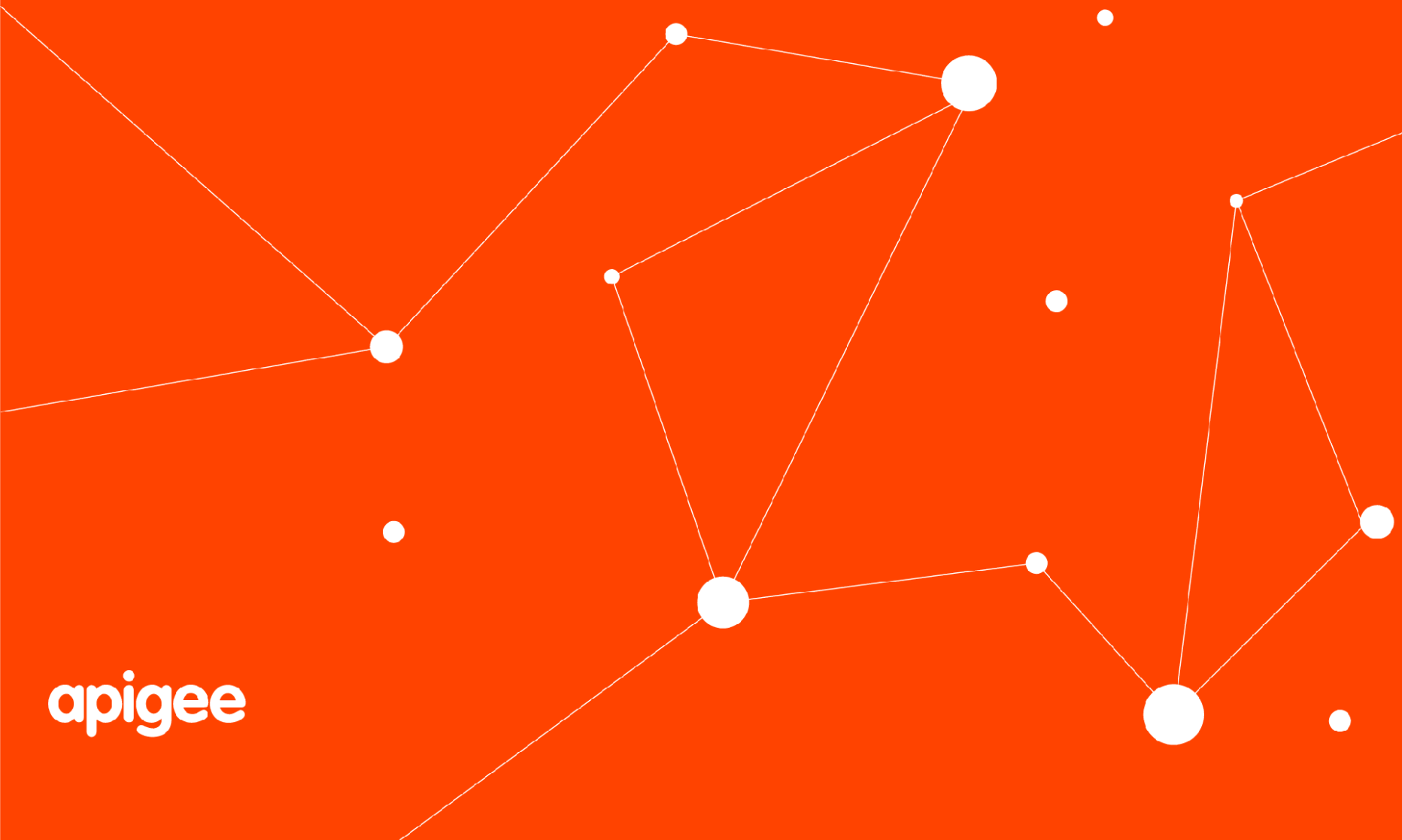
```
<MaskDataConfiguration name="default">
    <JSONPathsRequest>
        <JSONPathRequest>$.logonPassword</JSONPathRequest>
        <JSONPathRequest>$.cardNumber</JSONPathRequest>
    </JSONPathsRequest>
    <Variables>
        <Variable>password</Variable>
    </Variables>
    <XPathsRequest>
        <XPathRequest>/root/logonPassword</XPathRequest>
        <XPathRequest>/root/cardNumber</XPathRequest>
    </XPathsRequest>
</MaskDataConfiguration>
```

- Data is masked in trace

Request Content

Body	{"logonPassword":"**********","lastName":"Smith","firstName":"Bob"}

Thank you



apigee