Replay attacks on HTTP Signatures

draft-hoypat-httpbis-message-signatures-ekm

WEBBOTAUTH — IETF 124

Jonathan Hoyland, **Christopher Patton**

Background

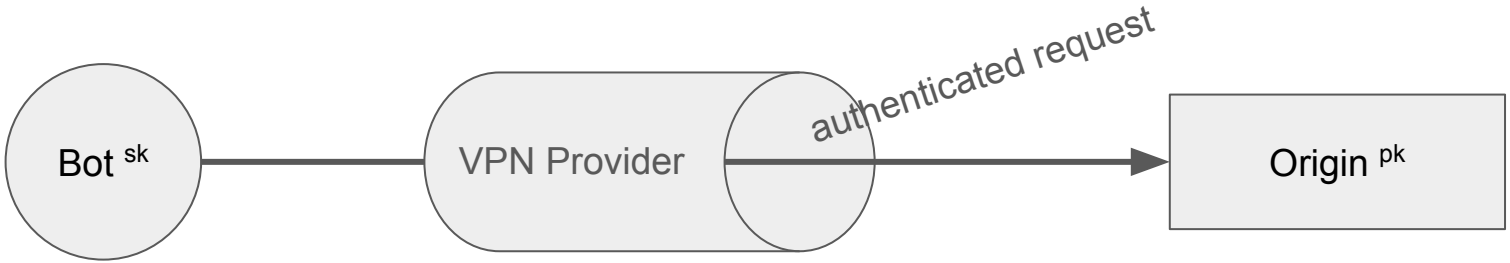
- draft-meunier-web-bot-auth-architecture uses HTTP Signatures (RFC 9421) to authenticate requests from registered bots.
    - Defines various ***components*** of the HTTP message that can be signed, e.g.: @method, @path, @query-param, etc.
    - Defines various ***parameters*** that can be bound to the signature, e.g.: key ID, nonce, validity window, etc.
- All information covered by the signature can be extracted from the bot's request. This implies the request can be ***replayed*** under certain conditions. To mitigate (RFC 9421, Section 7.2.2):
    - Sign more components and parameters that are likely to differ across requests
    - Use the ***nonce*** parameter

Replay attacks

Bot makes authenticated request to origin.

Replay attacks *by forward proxy*

Forward proxy between bot and origin (e.g., a VPN) can intercept authenticated requests …



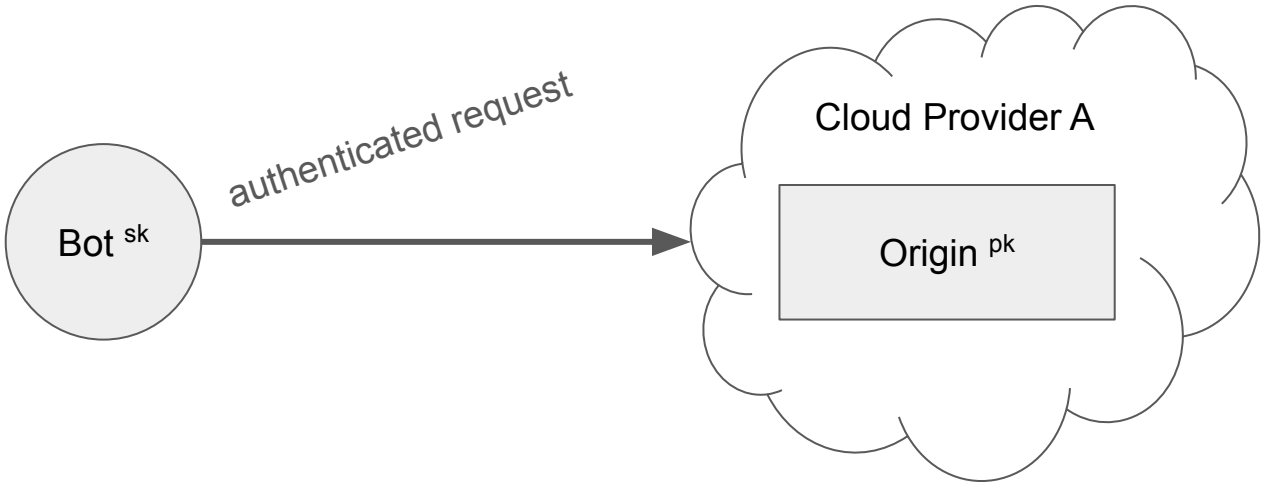
Replay attacks *by forward proxy*

… and later replay the request (impersonating the bot).

Fortunately, using a nonce prevents this attack 😌

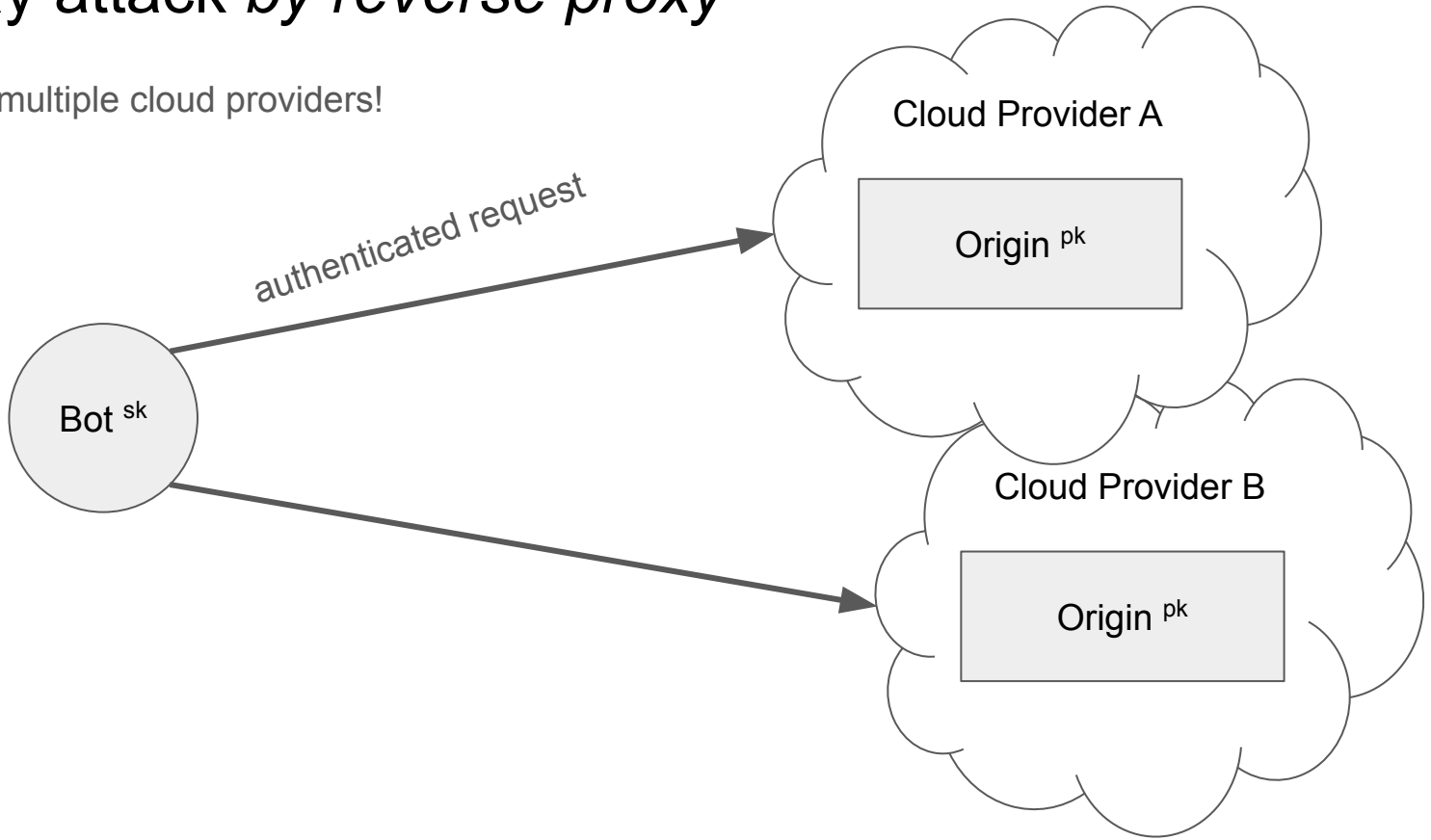
Replay attack *by reverse proxy*

Origin may be hosted on some cloud provider…



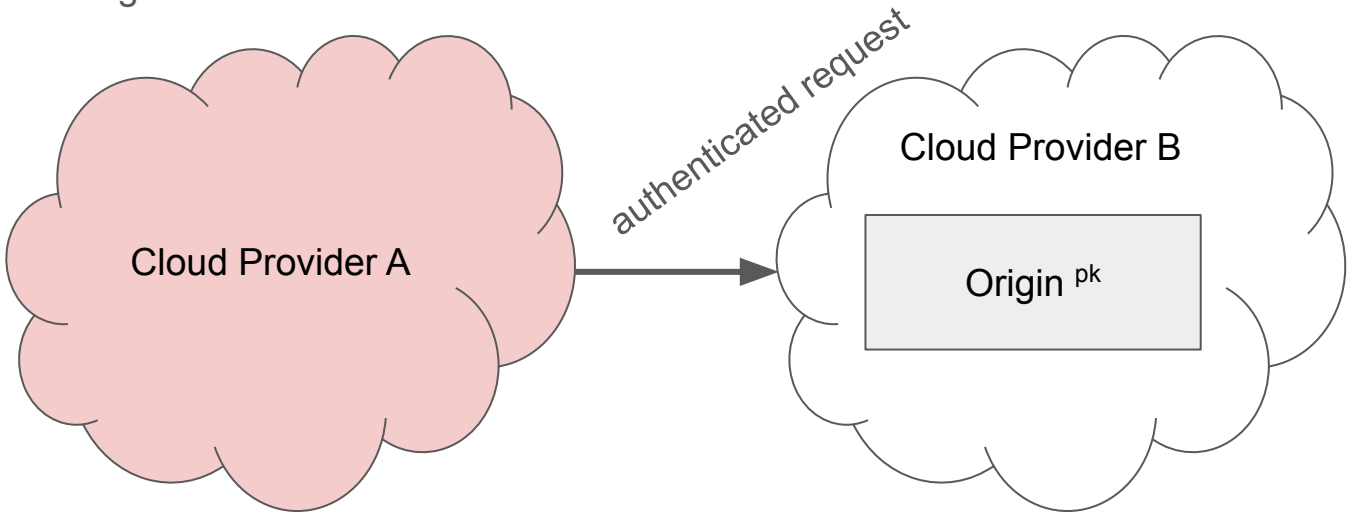
Replay attack *by reverse proxy*

… or on multiple cloud providers!



Replay attack *by reverse proxy*

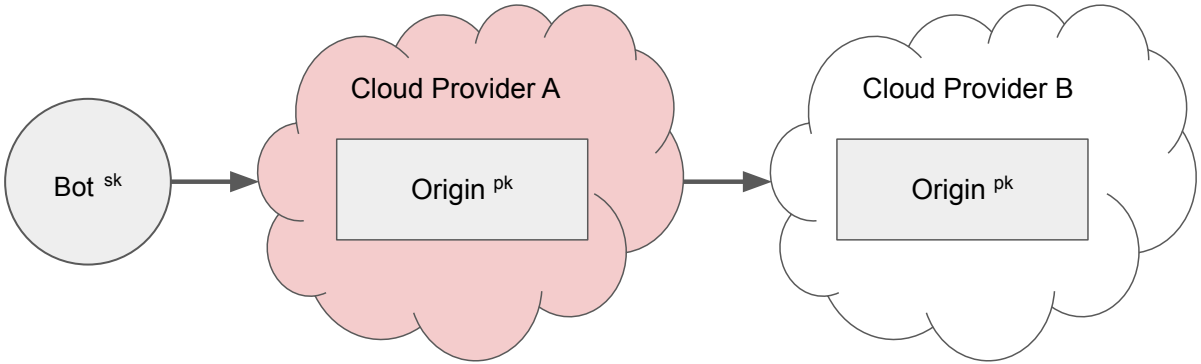
Any cloud provider can ***replay*** a request from the bot to any cloud provider used by the same origin.



Using a nonce doesn't help here: cloud providers don't share replay state (and Provider A is lying anyway).

Binding HTTP Signatures to TLS

- Goal of draft-hoypat-httpbis-message-signatures-ekm is to extend RFC 9421 to allow binding signature to the TLS channel.
  - Concretely: Client would export keying material (EKM) from the TLS connection state (a random, 32-byte string known only to the client and the server) and plumb it into the input to the HTTP signature.
  - Prevents replay by reverse proxy: EKM(Bot, Provider A) ≠ EKM(Provider A, Provider B) ⇒ signature validation will fail

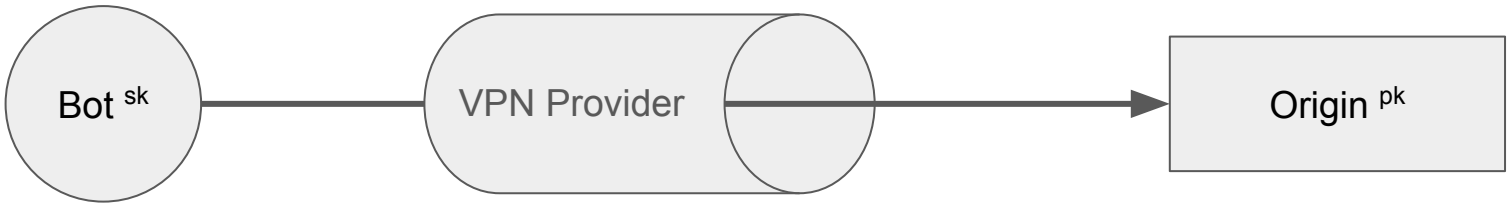


Alternatives

- RFC 9279 (The Concealed HTTP Authentication Scheme) signs a channel binding
  - Observation: you don't need to sign other properties if you sign a channel binding
- draft-rosomakho-tls-wimse-cert-hint prompts the TLS server to do TLS client authentication (client also indicates which roots the server should use to authenticate them)

Problem: Incompatible with forward proxies

EKM(Bot, VPN Provider) ≠ EKM(VPN Provider, Origin).



Need to convey EKM(Bot, VPN Provider) to Origin over a separate, secure channel. ***How!?***

Discussion

1. Do replay attacks matter?
    a. Hypothetically, if origin wanted to ***meter*** bots (e.g., charge $0.02/request), then a malicious cloud provider could rack up the bot's bill by replaying its requests.
2. For which uses cases Is forward proxy incompatibility a deal breaker?
    a. How to work around this?
3. Implementation complexity: Many HTTP stacks try abstract away from the user the details of the TLS connection state and even construction of HTTP requests.
4. How to negotiate the use of EKM in the HTTP signature?
    a. We won't always use EKM in every signature.
    b. Bots might try it with origins that don't support EKM yet.

Next steps

1. Get Feedback from HTTPBIS on the draft and maybe ask for adoption of draft-hoypat-httpbis-message-signatures-ekm.
2. Anything for WEBBOTAUTH to do?
3. Interested in implementation?