HTTP Message Signatures

Justin Richer & Annabelle Backman
IETF HTTP WG Interim
September 28, 2021
How HTTP Message Signing works

1. Choose covered portions and crypto parameters
2. Normalize the HTTP message components
3. Generate a signature input string
4. Sign the string creating a signature output
5. Add the signature output and parameters as structured HTTP headers
Example HTTP Message

POST /foo?param=value&pet=dog HTTP/1.1
Host: example.com
Date: Tue, 20 Apr 2021 02:07:55 GMT
Content-Type: application/json
Content-Length: 18

{"hello": "world"}
Sign These Components

POST /foo?param=value&pet=dog HTTP/1.1
Host: example.com
Date: Tue, 20 Apr 2021 02:07:55 GMT
Content-Type: application/json
Content-Length: 18

{"hello": "world"}
Message Component Identifiers

"@method"
POST /foo?param=value&pet=dog HTTP/1.1
Host: example.com
Date: Tue, 20 Apr 2021 02:07:55 GMT
Content-Type: application/json
Content-Length: 18

{"hello": "world"}
Signature Input Parameters

("@method" "@target-uri" "content-type");created=1618884475;keyid="test-key-1"

Signed Message Component Identifiers (ordered list)

Signature Creation Timestamp

Key Identifier
Signature Input Types

"@target-uri": https://example.com/foo?param=value&pet=dog
"content-type": application/json
Signature Input String

"@method": POST

Message Component Identifier
(structured field string)

Message Component Value
Signature Input String

"@method": POST
"@target-uri": https://example.com/foo?param=value&pet=dog

Message Component Identifier (structured field string)

Message Component Value
Signature Input String

"@method": POST
"@target-uri": https://example.com/foo?param=value&pet=dog
"content-type": application/json
Signature Input String

"@method": POST
"@target-uri": https://example.com/foo?param=value&pet=dog
"content-type": application/json
"@signature-params": (@method "@target-uri"
    "content-type");created=1618884475;keyid="test-key-1"

Message Component Identifier
(structured field string)

Message Component Value
(newline added for readability)
Signature Input String

"@method": POST
"@target-uri": https://example.com/foo?param=value&pet=dog
"content-type": application/json
"@signature-params": (@method @target-uri
   "content-type");created=1618884475;keyid="test-key-1"
Signature Bytes

Lu2cC2Ifw3hkpXt8iC9g78qppHzEUo7hPyeFmDNqkMe4AvPzhz8cRhI1+eIBisvM7ceDh40m0RmKjA5CUL5TFs9NuUHC0xuZZeiy5u7THftAZZU6LgwRynMuOZgJAYXYDsGBKfxRkoGKVVEX1lSGi7RVhYl/EgWCJzuIbJ9mLeRxzaXRr3pZXz5xRaXcsXItpsK3AnWYHoc6YAT9hP5M3oJPeb3KRHoLAn4nheC0kFoyLzRAf6/BNb4I7JhwqVZMZBlndnI/KBXoTK7rzYFdpX/Cbtwv+XHgli9Qthktw9hXC4Kv4lp2fCGSPJPHKeyrZ0rhCcfe++eJe0Ykm3FIw==
Signed Request

POST /foo?param=value&pet=dog HTTP/1.1
Host: example.com
Date: Tue, 20 Apr 2021 02:07:55 GMT
Content-Type: application/json
Content-Length: 18
Signature-Input: sig1="{@method" "@target-uri"
  "content-type"};created=1618884475;keyid="test-key-1"

Signature:
sig1=Lu2cC2Ifw3hkpXt8iC9g78qppHzEUo7hPyeFmDNqkMe4AvPzhz8cRhI1+eIBisvM7ceDh4Om0
RmKjA5CUL5TFs9NuUHC0xuZZeiy5u7THftAZZU6LgwRynMuOZgJAYXYDsGBKfxRkoGKVVEX1lSGi7RV
hY1/EgWCJzuIbJ9mLeRxzaXRr3pZXz5xRaXcsXItpsK3AnWYHoc6YAT9hP5M3oJPeb3KRHoLAN4nheC
OkFoyLzRAf6/BNb4I7JhwqVZMZBlndnI/KTBXoTK7rzYFdpX/Cbtwv+XHgli9QtHktw9hXC4Kv4lp2f
CGSPJPHKeyrZ0rhCcfe++eJe0Ykm3FIw==:

{"hello": "world"}
How HTTP Message Verification works

1. Read the Signature-Input and Signature header values
2. Validate covered portions and crypto parameters
3. Normalize the HTTP message components
4. Re-generate the signature input string
5. Verify the signature against the signature input string
Some important aspects

• Detached signature, not encapsulation
• Uses HTTP Structured Fields
• Allows multiple signatures on a message
• Can sign most HTTP parts
• Works for requests and responses
• Relatively robust against common changes
Since Last We Met

- Solidified algorithm and key selection
- Aligned with HTTP terminology
- Added new specialty components
- Signature negotiation (Accept-Signature)
- Dropped list prefix component indexing
- Expanded examples (especially responses)
Specialty Components

- @signature-params
- @method
- @target-uri
- @authority
- @scheme
- @request-target
- @path
- @query
- @query-params
- @status
- @request-response
Accept-Signature:

```
sig1=("@method" "@target-uri" "content-type");keyid="test-key-1"
```
Accept-Signature

Accept-Signature: sig1="@method" "@target-uri" "content-type");keyid="test-key-1"

Sign These Message Components

Call the results this

With these parameters
Current Status

- Core signature process is *still* stable
- Implementations in several languages
- Seeing more feedback from implementors of older specs (Cavage, OAuth PoP)
- Proposed as basis for new OAuth PoP spec
- Default signature method in GNAP
- Editors writing security and privacy considerations
DISCUSSION:
Security and Privacy Considerations
DISCUSSION:
Special case: cookies
DISCUSSION:

Special case: empty vs. not-present headers
DISCUSSION:
EdDSA Signing
DISCUSSION:
Relationship to WPACK / Signed Exchanges
DISCUSSION:
Implementations
Next Steps

• Branding and framing
  – Normalization is a bigger part than signing
  – It’s also about verifying signatures
• Guidance to developers on choosing security parameters for their applications
• IANA registry guidelines
• More examples! More code!