HTTP 検査インスペクタ

HTTP 検査インスペクタの概要

タイプ	インスペクタ（サービス）
使用方法	検査
インスタンス タイプ	マルチトン
その他のインスペクタが必要	`stream_tcp`
有効	`true`

Hypertext Transfer Protocol （HTTP）は、クライアントとサーバー間でハイパーメディア（オーディオ、ビデオ、画像、およびテキスト）の交換を可能にするアプリケーション層プロトコルです。HTTP は、信頼性の高いメッセージ送信を必要とするステートレスプロトコルです。クライアントとサーバー間の通信は、HTTP 要求と応答の形式で行われます。

HTTP サーバーは通常、TCP/IP を介したポート 80 を使用します。セキュアバージョンの HTTP（HTTP/TLS または HTTPS）はポート 443 を使用します。HTTP は、プロトコルでアクセス制御と認証メカニズムを定義します。

`http_inspect` インスペクタは、HTTP メッセージのプロトコルデータユニット（PDU）を検出して分析します。`http_inspect` は、TCP ストリームから TCP ペイロードを受信し、カプセル化された HTTP メッセージを確認します。

HTTP インスペクタは、次の HTTP メッセージセクションを検出できます。

- 要求行

- ステータス行
- ヘッダー
- Content-Length メッセージ本文（Content-length ヘッダーで定義されるメッセージ本文）
- チャンクされたメッセージ本文
- 前のメッセージ本文（Content-Length ヘッダーのないメッセージ本文）
- Trailers

`http_inspect` インスペクタは、すべての HTTP ヘッダーフィールドと HTTP URI のコンポーネントを検出して正規化します。`http_inspect` インスペクタは TCP ポートを正規化しません。

`http_inspect` インスペクタは、4 種類の URI を検出できます。

- アスタリスク（*）：正規化されていません
- 権限（Authority）： HTTP CONNECT メソッドで使用される URI
- 発信元（Origin）：スラッシュで始まる URI （スキームや権限の存在なし）
- 絶対（Absolute）：スキーム、ホスト、および絶対パスを含む URI。

HTTP URI には、次のものを含めることができます。

- スキーム（Scheme）（ftp、http、または https）
- ホスト（Host ）（サーバーのドメイン名）
- TCP ポート（TCP port）
- パス （Path）（ディレクトリとファイル）
- クエリ（Query）（要求パラメータ）
- フラグメント （Fragment）（ファイルの一部）

HTTP メッセージのセクションでアラートを発生させるように `http_inspect` インスペクタを設定できます。次に例を示します。

- HTTP 要求または応答の本文から読み取るバイト数を指定する
- JavaScript の検出と正規化を有効にする
- さまざまな種類のファイルの圧縮解除に対応する
- HTTP URI の復号化をカスタマイズする

（注） `http_inspect` インスペクタは、ストリーム TCP ペイロードを部分的に検査できます。

HTTP 検査インスペクタを設定するためのベストプラクティス

http_inspect インスペクタを構成するときは、次のベストプラクティスを考慮してください。

- HTTP トラフィックに大きなビデオファイルが含まれている場合は、`request_depth` パラメータと `response_depth` パラメータを設定します。
- HTTP URI 検査パラメータのデフォルト設定を使用します。

```
"utf8": "true"
"plus_to_space": "true"
"percent_u": "true"
"utf8_bare_byte": "true"
"iis_unicode": "true"
"iis_double_decode": "true"
```

HTTP 検査インスペクタのパラメータ

HTTP サービスの設定

`binder` インスペクタは、HTTP サービスの設定を定義します。詳細については、『バインダインスペクタの概要』を参照してください。

例：

```
[
    {
        "when": {
             "service": "http",
             "role": any
        },
        "use": {
            "type": "http_inspect"
        }
    }
]
```

request_depth

HTTP メッセージの要求本文から読み取るバイト数を指定します。

検査するバイト数に制限を設定しない場合は、`-1` を指定します。分析する HTTP 本文データの量を制限する場合は、`request_depth` パラメータと `response_depth` パラメータを指定することをお勧めします。

HTTP ヘッダーのみを検査するには、`request_depth` を `0` に設定します。

型：整数

有効な範囲：-1 ～ 9,007,199,254,740,992 (max53)

デフォルト値：-1

response_depth

HTTP メッセージの応答本文から読み取るバイト数を指定します。

検査するバイト数に制限を設定しない場合は、-1 を指定します。分析する HTTP 本文データの量を制限する場合は、request_depth パラメータと response_depth パラメータを指定することをお勧めします。

HTTP ヘッダーのみを検査するには、response_depth を 0 に設定します。

型：整数

有効な範囲：-1 ～ 9,007,199,254,740,992 (max53)

デフォルト値：-1

unzip

gzip ファイルの圧縮を解除し、メッセージ本文を検査する前にデフレートするかどうかを指定します。圧縮解除を無効にすると、HTTP インスペクタは HTTP メッセージ本文のすべての部分を処理できなくなります。http_inspect インスペクタは HTTP ヘッダーを処理できます。

型：ブール値

有効な値：true、false

デフォルト値：true

maximum_host_length

Host HTTP ヘッダーフィールドで許可されるバイトの最大数を指定します。

ヘッダー値の長さに制限を設定しない場合は、-1 を指定します。

型：整数

有効な範囲：-1 ～ 9,007,199,254,740,992 (max53)

デフォルト値：-1

maximum_chunk_length

HTTP メッセージの本文チャンクで許可される最大バイト数を指定します。

HTTP チャンクのバイト数に制限を設定しない場合は、-1 を指定します。

型：整数

有効な範囲：-1 ～ 9,007,199,254,740,992 (max53)

デフォルト値：-1

normalize_utf

HTTP応答本文で検出されたUTFエンコーディング（UTF-8、UTF-7、UTF-16LE、UTF-16BE、UTF-32LE、およびUTF-32BE）を正規化するかどうかを指定します。`http_inspect`インスペクタは、HTTP `Content-Type`ヘッダーからのUTF文字エンコードを決定します。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`true`

decompress_pdf

HTTP応答本文で検出された`application/pdf`（PDF）ファイルのデフレート互換の圧縮部分の圧縮を解除するかどうかを指定します。`http_inspect`インスペクタは、`/FlateDecode`ストリームフィルタを使用してPDFファイルの圧縮を解除します。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

decompress_swf

HTTP応答本文で検出された`application/vnd.adobe.flash-movie`（SWF）ファイルの圧縮を解除するかどうかを指定します。

（注） HTTP GET応答で検出されたファイルの圧縮部分のみを圧縮解除できます。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

decompress_vba

HTTP応答本文で検出されたMicrosoft Office Visual Basic for Applicationsのマクロファイルの圧縮を解除するかどうかを指定します。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

decompress_zip

HTTP応答本文で検出された`application/zip`（ZIP）ファイルの圧縮を解除するかどうかを指定します。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

script_detection

スクリプトの末尾要素（<\script>）を検出した後、JavaScriptの内容を検査するかどうかを指定します。`http_inspect`は、スクリプトの末尾を検出すると、早期検出のために部分的に読み取られたメッセージ本文をすぐに転送します。スクリプト検出により、Snortは悪意のあるJavaScriptを含む可能性のある応答メッセージをすばやくブロックできます。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

normalize_javascript

HTTP 応答本文で JavaScript を正規化するためにレガシーメカニズムを使用するかどうかを指定します。このオプションは、レガシー JavaScript ノーマライザを構成します。`http_inspect` インスペクタは `unescape` 関数や `decodeURI` 関数、`String.fromCharCode` メソッドなどの難読化 JavaScript データを正規化します。HTTP インスペクタは、unescape 関数、decodeURI 関数、および decodeURIComponent 関数内のエンコード（%XX、%uXXXX、XX、および uXXXXi）を正規化します。

`http_inspect` インスペクタは連続する空白文字を検出し、1 つのスペースに正規化します。`normalize_javascript` が有効になっている場合、`max_javascript_whitespaces` を設定して、難読化された JavaScript の連続する空白文字の数を制限できます。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

js_norm_bytes_depth

正規化する入力 JavaScript のバイト数を指定します。このオプションは、拡張 JavaScript ノーマライザに固有です。

（注） 拡張 JavaScript ノーマライザを使用する場合、Lightweight Security Package（LSP）および Snort 3 のデフォルト設定が使用されます。JavaScript 固有の構成は、ネットワーク分析ポリシー（NAP）ユーザーインターフェイスからブロックされます。デフォルト設定を上書きしてノーマライザ設定をカスタマイズするには、`/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy`にある`NAPOverride.lua`ファイルを変更します。

http_inspect インスペクタは連続する空白文字を検出し、1 つのスペースに正規化します。インスペクタは、トラフィックを効果的に正規化するために、開始 <script> is in one PDU and the end </script> が別の PDU にあるさまざまな PDU のスクリプトを追跡します。新しいバッファ js_data は、ジャストインタイム（JIT）アプローチを使用して JavaScript コードを検出および正規化する Snort 3 IPS バッファに追加され、このオプションがルールで使用されている場合にのみノーマライザが呼び出されます。

http_inspect インスペクタは、JavaScript コードに関連付けられた関数名、変数名、およびラベル名を正規化します。さらに、インスペクタは、application/javascript または類似の MIME タイプを使用して、外部スクリプトの形式で転送される JavaScript コードを正規化します。ノーマライザは、JavaScript 機能がクライアント側からの元の入力から変更されない自動セミコロン挿入を実行します。

http_inspect インスペクタは、unescape 関数、decodeURI 関数、および decodeURIComponent 関数と、String.fromCharCode メソッドおよび String.fromCodePoint メソッドなどの難読化 JavaScript データを正規化します。HTTP インスペクタは、unescape 関数、decodeURI 関数、および decodeURIComponent 関数内のエンコード（%XX、%uXXXX、\uXX、\u{XXXX}\xXX、10 進数コードポイント、16 進数コードポイント）を正規化します。

また、http_inspect インスペクタは Javascript のプラス（+）演算子を正規化し、その演算子を使用して文字列を連結します。

JavaScript のバイト数に制限を設定しない場合は、-1 を指定します。

型：整数

有効な範囲：-1 ～ 9,007,199,254,740,992 (max53)

デフォルト値：-1

js_norm_identifier_depth

正規化する一意の JavaScript 識別子の最大数を指定します。このオプションは、拡張 JavaScript ノーマライザに固有です。

（注） 拡張 JavaScript ノーマライザを使用する場合、Lightweight Security Package（LSP）および Snort 3 のデフォルト設定が使用されます。JavaScript 固有の構成は、ネットワーク分析ポリシー（NAP）ユーザーインターフェイスからブロックされます。デフォルト設定を上書きしてノーマライザ設定をカスタマイズするには、/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy にある NAPOverride.lua ファイルを変更します。

型：整数

有効な範囲：0 ～ 65536

デフォルト値：65536

js_norm_max_bracket_depth

正規化する JavaScript ブラケットのネストの最大深度を指定します。このオプションは、拡張 JavaScript ノーマライザに固有です。

（注） 拡張 JavaScript ノーマライザを使用する場合、Lightweight Security Package（LSP）および Snort 3 のデフォルト設定が使用されます。JavaScript 固有の構成は、ネットワーク分析ポリシー（NAP）ユーザーインターフェイスからブロックされます。デフォルト設定を上書きしてノーマライザ設定をカスタマイズするには、`/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy` にある `NAPOverride.lua` ファイルを変更します。

型：整数

有効な範囲：`1` ～ `65535`

デフォルト値：`256`

js_norm_max_scope_depth

正規化する JavaScript スコープのネストの最大深度を指定します。このオプションは、拡張 JavaScript ノーマライザに固有です。

（注） 拡張 JavaScript ノーマライザを使用する場合、Lightweight Security Package（LSP）および Snort 3 のデフォルト設定が使用されます。JavaScript 固有の構成は、ネットワーク分析ポリシー（NAP）ユーザーインターフェイスからブロックされます。デフォルト設定を上書きしてノーマライザ設定をカスタマイズするには、`/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy` にある `NAPOverride.lua` ファイルを変更します。

型：整数

有効な範囲：`1` ～ `65535`

デフォルト値：`256`

js_norm_max_tmpl_nest

正規化する JavaScript テンプレートリテラルのネストの最大深度を指定します。このオプションは、拡張 JavaScript ノーマライザに固有です。

（注） 拡張 JavaScript ノーマライザを使用する場合、Lightweight Security Package（LSP）および Snort 3 のデフォルト設定が使用されます。JavaScript 固有の構成は、ネットワーク分析ポリシー（NAP）ユーザーインターフェイスからブロックされます。デフォルト設定を上書きしてノーマライザ設定をカスタマイズするには、`/ftd/app_data/Volume/root1/ngfw/var/cisco/deploy` にある `NAPOverride.lua` ファイルを変更します。

型：整数

有効な範囲：`0` 〜 `255`

デフォルト値：`32`

max_javascript_whitespaces

JavaScript 難読化データ内で許可される連続する空白文字の最大数を指定します。

型：整数

有効な範囲：`1` 〜 `65535`

デフォルト値：`200`

percent_u

`%uNNNN` エンコーディングと `%UNNNN` エンコーディングを正規化するかどうかを指定します。4 つの `N` 文字は、Microsoft インターネット インフォメーション サービス（IIS）の Unicode コードポイントに関連する 16 進数でエンコードされた値を表します。正規の クライアントが `%u` エンコーディングを使用することはほとんどないため、`%u` エンコーディングによってエンコードされている HTTP トラフィックを復号化することをお勧めします。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

utf8

URI の標準 UTF-8 Unicode シーケンスを正規化するかどうかを指定します。`http_inspect` インスペクタは、2 バイトまたは 3 バイトの UTF-8 文字を 1 バイトに正規化できます。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`true`

utf8_bare_byte

URL またはパーセントエンコードされていないバイトを含む UTF-8 文字を正規化するかどうかを指定します。`utf8_bare_byte` パラメータを有効にすることをお勧めします。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

iis_unicode

HTTP メッセージ内の文字を Unicode コードポイントで正規化するかどうかを指定します。

（注） `iis_unicode` パラメータを有効にすることをお勧めします。Unicode は、攻撃試行や回避試行によく見られます。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`false`

iis_unicode_code_page

IIS Unicode マップファイルのコードページを使用するかどうかを指定します。

型：整数

有効な範囲：`1` ～ `65535`

デフォルト値：`1252`

iis_double_decode

URL でエンコードされた文字を二重デコードして文字を正規化するかどうかを指定します。要求 URI を介して 2 回通過して IIS 二重エンコードトラフィックを復号化します。`iis_double_decode` パラメータを有効にすることをお勧めします。通常、二重エンコーディングは攻撃シナリオでのみ見られます。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`true`

oversize_dir_length

URL ディレクトリで許可される最大バイト数を指定します。

型：整数

有効な範囲：`1` ～ `65535`

デフォルト値：`300`

backslash_to_slash

URI でバックスラッシュ（\）をスラッシュ（/）に置き換えるかどうかを指定します。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`true`

plus_to_space

URI の プラス記号（+）を <sp> に置き換えるかどうかを指定します。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`true`

simplify_path

URIディレクトリパスを最も単純な形式に減らすかどうかを指定します。余分なトラバーサルを含む URI ディレクトリ パスには、. 、.. 、/ が含まれる場合があります。

型：ブール値

有効な値：`true`、`false`

デフォルト値：`true`

xff_headers

調べる `X-Forwarded-For` HTTP ヘッダーのタイプを指定します。`xff_headers` パラメータで、`X-Forwarded-For` ヘッダーを優先順位の高いものから順にリストします。

カスタム `X-Forwarded-For` タイプのヘッダーを定義できます。元のクライアント IP アドレスを伝送する HTTP ヘッダーには、ベンダー固有のヘッダー名を付けることができます。このシナリオでは、`xff_headers` パラメータは、カスタムヘッダーを HTTP インスペクタに導入する方法を提供します。

`xff_headers` のデフォルト値は、2 つの一般的に知られているヘッダーである `x-forwarded-for` `true-client-ip` です。ストリーム内に両方のデフォルトヘッダーが存在する場合、`true-client-ip` よりも `x-forwarded-for` が優先されます。`X-Forwarded-For` HTTP ヘッダーを複数指定する場合は、ヘッダー名をスペースで区切ります。

型：文字列

有効な値：`x-forwarded-for`、`true-client-ip`

デフォルト値：`x-forwarded-for true-client-ip`

HTTP 検査インスペクタのルール

`http_inspect` インスペクタルールを有効にし、イベントを生成し、インライン展開では、違反パケットをドロップします。。

表 *1: HTTP* 検査インスペクタのルール

GID:SID	ルール メッセージ
119:1	URIに予約されていない文字のパーセントエンコーディングがある（URI has percent-encoding of an unreserved character）
119:2	URI はパーセントエンコーディングされており、結果は再びパーセントエンコーディングされる（URI is percent encoded and the result is percent encoded again）
119:3	URI に標準以外の %u スタイルの Unicode エンコーディングがある（URI has non-standard %u-style Unicode encoding）
119:4	URIに、パーセントエンコーディングされていないバイトを含むUnicodeエンコーディングがある（URI has Unicode encodings containing bytes that were not percent-encoded）
119:6	URI に 2 バイトまたは 3 バイトの UTF-8 エンコーディングがある（URI has two-byte or three-byte UTF-8 encoding）
119:7	URI に Unicode マップ コード ポイント エンコーディングがある（URI has unicode map code point encoding）
119:8	URI パスに連続したスラッシュ文字が含まれている（URI path contains consecutive slash characters）
119:9	URIのパス部分にバックスラッシュ文字が表示される（backslash character appears in the path portion of a URI）
119:10	URI パスに現在のディレクトリを繰り返す /./ パターンが含まれている（URI path contains /./ pattern repeating the current directory）
119:11	URIパスにディレクトリを上に移動する/../パターンが含まれている（URI path contains /../ pattern moving up a directory）
119:12	HTTP 開始行内にタブ文字がある（Tab character in HTTP start line）
119:13	HTTP 開始行またはヘッダー行が CR なしに LF で終了する（HTTP start line or header line terminated by LF without a CR）
119:14	正規化された URI に bad_characters リストの文字が含まれている（Normalized URI includes character from bad_characters list）

GID:SID	ルール メッセージ
119:15	URI パスに oversize_dir_length パラメータよりも長いセグメントが含まれている（URI path contains a segment that is longer than the oversize_dir_length parametery）
119:16	チャンク長が、設定された maximum_chunk_length を超えている（chunk length exceeds configured maximum_chunk_length）
119:18	URI パスにルートディレクトリを超える /../ が含まれている（URI path includes /../ that goes above the root directory）
119:19	HTTP ヘッダー行が 4096 バイトを超えている（HTTP header line exceeds 4096 bytes）
119:20	HTTP メッセージに 200 を超えるヘッダーフィールドがある（HTTP message has more than 200 header fields）
119:21	HTTP メッセージに複数の Content-Length ヘッダー値がある（HTTP message has more than one Content-Length header value）
119:24	ホストヘッダーフィールドが複数回表示されるか、複数の値が含まれている（Host header field appears more than once or has multiple values）
119:25	HTTP ホストヘッダーフィールド値の長さが maximum_host_length オプションを超えている（length of HTTP Host header field value exceeds maximum_host_length option）
119:28	content-length またはチャンクのない HTTP POST 要求または PUT 要求（HTTP POST or PUT request without content-length or chunks）
119:31	Snort が HTTP 要求メソッドを認識していない（HTTP request method is not known to Snort）
119:32	HTTP 要求は HTTP/0.9 と呼ばれるプリミティブ HTTP フォーマットを使用する（HTTP request uses primitive HTTP format known as HTTP/0.9）
119:33	HTTP 要求 URI にパーセントエンコーディングされていない余白文字が含まれている（HTTP request URI has space character that is not percent-encoded）
119:34	HTTP 接続に応答されていない 100 を超える同時パイプライン要求がある（HTTP connection has more than 100 simultaneous pipelined requests that have not been answered）
119:102	HTTP 応答の無効なステータスコード（invalid status code in HTTP response）
119:104	HTTP 応答に正規化に失敗した UTF 文字セットが含まれている（HTTP response has UTF character set that failed to normalize）

GID:SID	ルール メッセージ
119:105	HTTP 応答に UTF-7 文字セットが含まれている（HTTP response has UTF-7 character set）
119:109	JavaScript の難読化レベルの幅がある（more than one level of JavaScript obfuscation）
119:110	連続した JavaScript の空白が最大許容数を超えている（consecutive JavaScript whitespaces exceed maximum allowed）
119:111	JavaScript 難読化データ内にエンコーディングが複数ある（multiple encodings within JavaScript obfuscated data）
119:112	SWF ファイルの zlib の圧縮解除に失敗した（SWF file zlib decompression failure）
119:113	SWF ファイル LZMA の圧縮解除に失敗した（SWF file LZMA decompression failure）
119:114	PDF ファイルのデフレートに失敗した（PDF file deflate decompression failure）
119:115	PDF ファイルのサポート対象外の圧縮タイプ（PDF file unsupported compression type）
119:116	複数の圧縮が PDF ファイルに適用されている（PDF file with more than one compression applied）
119:117	PDF ファイルの解析に失敗した（PDF file parse failure）
119:201	HTTP トラフィックではないか、または回復不能な HTTP プロトコルエラー（not HTTP traffic or unrecoverable HTTP protocol error）
119:202	チャンクの長さの先行ゼロが多すぎる（chunk length has excessive leading zeros）
119:203	HTTP メッセージの前または間の空白文字（white space before or between HTTP messages）
119:204	URI のない要求メッセージ（request message without URI）
119:205	HTTP 応答の理由句内の制御文字（control character in HTTP response reason phrase）
119:206	開始行の不正で余分な空白文字（illegal extra whitespace in start line）
119:207	HTTP バージョンの破損（corrupted HTTP version）
119:209	HTTP ヘッダーでのフォーマットエラー（format error in HTTP header）

GID:SID	ルール メッセージ
119:210	チャンクヘッダーのオプションが存在する（chunk header options present）
119:211	URI の形式が正しくない（URI badly formatted）
119:212	URI のパーセントエンコーディングのタイプが認識されない（unrecognized type of percent encoding in URI）
119:213	HTTP チャンクのフォーマットが正しくない（HTTP chunk misformatted）
119:214	チャンク長に隣接する空白文字がある（white space adjacent to chunk length）
119:215	ヘッダー名内に空白文字がある（white space within header name）
119:216	過度な gzip 圧縮（excessive gzip compression）
119:217	gzip 圧縮解除に失敗した（gzip decompression failed）
119:218	HTTP 0.9 が要求され、その後に別の要求が続いている（HTTP 0.9 requested followed by another request）
119:219	通常の要求の後に HTTP 0.9 要求が続いている（HTTP 0.9 request following a normal request）
119:220	メッセージに Content-Length と Transfer-Encoding の両方がある（message has both Content-Length and Transfer-Encoding）
119:221	Transfer-Encoding またはゼロ以外の Content-Length と組み合わされた本文がないことを示すステータスコード（status code implying no body combined with Transfer-Encoding or nonzero Content-Length）
119:222	Transfer-Encoding の末尾がチャンクされていない（Transfer-Encoding not ending with chunked）
119:223	チャンク前のエンコーディングによる Transfer-Encoding（Transfer-Encoding with encodings before chunked）
119:224	HTTP トラフィックの形式が誤っている（misformatted HTTP traffic）
119:225	サポート対象外の Content-Encoding が使用されている（unsupported Content-Encoding used）
119:226	不明な Content-Encoding が使用されている（unknown Content-Encoding used）
119:227	複数の Content-Encodings が適用されている（multiple Content-Encodings applied）
119:228	クライアント要求前のサーバー応答（server response before client request）

GID:SID	ルール メッセージ
119:229	サーバー応答の PDF/SWF/ZIP 圧縮解除が大きすぎる（PDF/SWF/ZIP decompression of server response too big）
119:230	HTTP メッセージヘッダー名の非表示文字（nonprinting character in HTTP message header name）
119:231	HTTP ヘッダーの Content-Length 値が正しくない（bad Content-Length value in HTTP header）
119:232	HTTP ヘッダー行の折り返し（HTTP header line wrapped）
119:233	HTTP ヘッダー行が LF のない CR で終了した（HTTP header line terminated by CR without a LF）
119:234	チャンクが非標準セパレータで終了した（chunk terminated by nonstandard separator）
119:235	チャンクの長さが CR なしの LF で終了した（chunk length terminated by LF without CR）
119:236	複数の応答に 100 番台のステータスコードがある（chunk length terminated by LF without CR）
119:237	100 番台のステータスコードが Expect ヘッダーに対応していない（100 status code not in response to Expect header）
119:238	100 または 101 以外の 1XX ステータスコード（1XX status code other than 100 or 101）
119:239	メッセージの本文なしで送信されるヘッダーを予測（Expect header sent without a message body）
119:240	HTTP 1.0 メッセージに Transfer-Encoding ヘッダーが含まれている（HTTP 1.0 message with Transfer-Encoding header）
119:241	Content-Transfer-Encoding が HTTP ヘッダーとして使用されている（Content-Transfer-Encoding used as HTTP header）
119:242	チャンクされたメッセージトレーラに不正なフィールドがある（illegal field in chunked message trailers）
119:243	ヘッダーフィールドが不適切に2回表示されるか、2つの値を持つ（header field inappropriately appears twice or has two values）
119:244	Content-Encoding ヘッダーでチャンクされた無効な値
119:245	Range ヘッダーのない要求に 206 応答が送信された（206 response sent to a request without a Range header）

GID:SID	ルール メッセージ
119:246	バージョンフィールドの HTTP がすべて大文字ではない（HTTP in version field not all upper case）
119:247	重要なヘッダー値に空白文字が埋め込まれている（white space embedded in critical header value）
119:248	gzip 圧縮データの後に予期しない非 gzip データが続く（gzip compressed data followed by unexpected non-gzip data）
119:249	過剰な HTTP パラメータキーが繰り返される（excessive HTTP parameter key repeats）
119:253	メッセージの本文に HTTP CONNECT 要求が含まれている（HTTP CONNECT request with a message body）
119:254	CONNECT 要求後、CONNECT 応答前の HTTP クライアントからサーバーへのトラフィック（HTTP client-to-server traffic after CONNECT request but before CONNECT response）
119:255	HTTP CONNECT 2XX 応答にContent-Length ヘッダーが含まれている（HTTP CONNECT 2XX response with Content-Length header）
119:256	HTTP CONNECT 2XX 応答に Transfer-Encoding ヘッダーが含まれている（HTTP CONNECT 2XX response with Transfer-Encoding header）
119:257	HTTP CONNECT 応答に 1XX ステータスコードが含まれている（HTTP CONNECT response with 1XX status code）
119:258	要求メッセージが完了する前の HTTP CONNECT 応答（HTTP CONNECT response before request message completed）
119:259	HTTP Content-Disposition ファイル名パラメータの形式が間違っている（malformed HTTP Content-Disposition filename parameter）
119:260	HTTP Content-Length メッセージの本文が切り捨てられた（HTTP Content-Length message body was truncated）
119:261	HTTP チャンクメッセージの本文が切り捨てられた（HTTP chunked message body was truncated）
119:262	HTTP URI スキーム長が 10 文字を超えている（HTTP URI scheme longer than 10 characters）
119:263	HTTP/1 クライアントが HTTP/2 へのアップグレードを要求した（HTTP/1 client requested HTTP/2 upgrade）
119:264	HTTP/1 サーバーが HTTP/2 へのアップグレードを許可した（HTTP/1 server granted HTTP/2 upgrade）

GID:SID	ルール メッセージ
119:265	JavaScript に不正なトークンがある（bad token in JavaScript）
119:266	JavaScript に予期しないスクリプト開始タグがある（unexpected script opening tag in JavaScript）
119:267	JavaScript に予期しないスクリプト終了タグがある（unexpected script closing tag in JavaScript）
119:268	外部スクリプトタグの下に JavaScript コードがある（JavaScript code under the external script tags）
119:269	短い形式のスクリプト開始タグ（script opening tag in a short form）
119:270	一意の JavaScript 識別子の最大数（max number of unique JavaScript identifiers reached）
119:271	JavaScript ブラケットのネストがキャパシティを超えている（JavaScript bracket nesting is over capacity）
119:272	HTTP Accept-Encoding ヘッダーに連続したコンマがある（Consecutive commas in HTTP Accept-Encoding header）
119:273	JavaScript の正規化中に PDU が欠落した（missed PDUs during JavaScript normalization）
119:274	JavaScript スコープのネストがキャパシティを超えている（JavaScript scope nesting is over capacity）
119:275	HTTP/1 バージョンが 1.0 または 1.1e 以外（HTTP/1 version other than 1.0 or 1.1e）
119:276	開始行の HTTP バージョンが 0 になっている（HTTP version in start line is 0）
119:277	開始行の HTTP バージョンが 1 より大きい（HTTP version in start line is higher than 1）

HTTP 検査インスペクタの侵入ルールのオプション

http_client_body

検出カーソルを HTTP リクエストの本文に設定します。HTTP メッセージで HTTP ヘッダーを指定しなかった場合、Snort は URI 正規化を使用して `http_client_body` を正規化します。URI 正規化は通常、`http_header` に適用されます。

シンタックス：`http_client_body;`

例：`http_client_body;`

http_cookie

抽出された HTTP `Cookie` ヘッダーフィールドに検出カーソルを設定します。`http_cookie` ルールオプションには、パラメータの `http_cookie.request`、`http_cookie.with_header`、`http_cookie.with_body`、および `http_cookie.with_trailer` が含まれます。

シンタックス：http_cookie: <parameter>, <parameter>

例：`http_cookie: request;`

http_cookie.request

HTTP 要求メッセージで検出された HTTP Cookie を照合します。HTTP 応答を確認する場合は、HTTP 要求 Cookie を使用します。`http_cookie.request` パラメータはオプションです。

シンタックス：`http_cookie: request;`

例：`http_cookie: request;`

http_cookie.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。`http_cookie.with_header` パラメータはオプションです。

シンタックス：`http_cookie: with_header;`

例：`http_cookie: with_header;`

http_cookie.with_body

ルールの別の部分で確認するのは `http_cookie` ルールオプションではなく、HTTP メッセージの本文であることを指定します。`http_cookie.with_body` パラメータはオプションです。

シンタックス：`http_cookie: with_body;`

例：`http_cookie: with_body;`

http_cookie.with_trailer

ルールの別の部分で確認するのは `http_cookie` ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。`http_cookie.with_trailer` パラメータはオプションです。

シンタックス：`http_cookie: with_trailer;`

例：`http_cookie: with_trailer;`

http_header

検出カーソルを正規化された HTTP ヘッダーに設定します。個々のヘッダー名を指定するには、`field` オプションを使用します。

http_header ルールオプションには、パラメータの http_header.field、http_header.request、http_header.with_header、http_header.with_body、および http_header.with_trailer が含まれます。

シンタックス：http_header: field <field_name>,<parameter>, <parameter>

例：http_header: field Content-Type, with_trailer;

http_header.field

指定したヘッダー名を正規化された HTTP ヘッダーと照合します。ヘッダー名では大文字と小文字が区別されません。ヘッダー名を指定しなかった場合、HTTP インスペクタは、HTTP Cookie のヘッダー（Cookie と Set-Cookie）を除くすべてのヘッダーを確認します。

型：文字列

シンタックス：http_header: field <field_name>;

有効な値：HTTP ヘッダー名。

例：http_header: field Content-Type;

http_header.request

HTTP 要求で検出されたヘッダーを照合します。HTTP 応答を確認する場合は、HTTP 要求ヘッダーを使用します。http_header.request パラメータはオプションです。

シンタックス：http_header: request;

例：http_header: request;

http_header.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。http_header.with_header パラメータはオプションです。

シンタックス：http_header: with_header;

例：http_header: with_header;

http_header.with_body

ルールの別の部分で確認するのは http_header ルールオプションではなく、HTTP メッセージの本文であることを指定します。http_header.with_body パラメータはオプションです。

シンタックス：http_header: with_body;

例：http_header: with_body;

http_header.with_trailer

ルールの別の部分で確認するのは http_header ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。http_header.with_trailer パラメータはオプションです。

シンタックス：`http_header: with_trailer;`

例：`http_header: with_trailer;`

http_method

検出カーソルを HTTP 要求のメソッドに設定します。一般的な HTTP 要求メソッドの値は、`GET`、`POST`、`OPTIONS`、`HEAD`、`DELETE`、`PUT`、`TRACE`、および `CONNECT` です。

`http_method` ルールオプションには、パラメータの `http_method.with_header`、`http_method.with_body`、および `http_method.with_trailer` が含まれます。

シンタックス：`http_method: <parameter>, <parameter>;`

例：`http_method; content:"GET";`

http_method.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。`http_method.with_header` パラメータはオプションです。

シンタックス：`http_method: with_header;`

例：`http_method: with_header;`

http_method.with_body

ルールの別の部分で確認するのは `http_header` ルールオプションではなく、HTTP メッセージの本文であることを指定します。`http_method.with_body` パラメータはオプションです。

シンタックス：`http_method: with_body;`

例：`http_method: with_body;`

http_method.with_trailer

ルールの別の部分で確認するのは `http_header` ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。`http_method.with_trailer` パラメータはオプションです。

シンタックス：`http_method: with_trailer;`

例：`http_method: with_trailer;`

http_param

検出カーソルを指定した HTTP パラメータキーに設定します。HTTP パラメータキーは、クエリまたは要求の本文に表示される場合があります。

`http_param` ルールオプションには、`http_param.param` パラメータと `http_method.nocase` が含まれます。

シンタックス：`http_param: <parameter_key>, nocase;`

例：`http_param: offset, nocase;`

http_param.param

指定したパラメータを照合します。

型：文字列

シンタックス：`http_param: <http_parameter>;`

有効な値：要求クエリパラメータまたは要求の本文フィールド。

例：`http_param: offset;`

http_param.nocase

指定したパラメータを照合しますが、大文字小文字は考慮されません。`http_param.nocase` パラメータはオプションです。

シンタックス：`http_param: nocase;`

例：`http_param: nocase;`

http_raw_body

検出カーソルを正規化されていない要求または応答メッセージの本文に設定します。

シンタックス：`http_raw_body;`

例： `http_raw_body;`

http_raw_cookie

検出カーソルを正規化されていない HTTP `Cookie` ヘッダーに設定します。`http_raw_cookie` ルールオプションには、パラメータの `http_raw_cookie.request`、`http_raw_cookie.with_header`、`http_raw_cookie.with_body`、および `http_raw_cookie.with_trailer` が含まれます。

シンタックス：`http_raw_cookie: <parameter>, <parameter>;`

例：`http_raw_cookie: request;`

http_raw_cookie.request

HTTP 要求で検出された Cookie を照合します。応答メッセージを確認する場合は、HTTP 要求の Cookie を使用します。`http_raw_cookie.request` パラメータはオプションです。

シンタックス：`http_raw_cookie: request;`

例：`http_raw_cookie: request;`

http_raw_cookie.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。`http_raw_cookie.with_header` パラメータはオプションです。

シンタックス：`http_raw_cookie: with_header;`

例：`http_raw_cookie: with_header;`

http_raw_cookie.with_body

ルールの別の部分で確認するのは`http_raw_cookie`ルールオプションではなく、HTTPメッセージの本文であることを指定します。`http_raw_cookie.with_body`パラメータはオプションです。

シンタックス：`http_raw_cookie: with_body;`

例：`http_raw_cookie: with_body;`

http_raw_cookie.with_trailer

ルールの別の部分で確認するのは`http_raw_cookie`ルールオプションではなく、HTTPメッセージのトレーラであることを指定します。`http_raw_cookie.with_trailer`パラメータはオプションです。

シンタックス：`http_raw_cookie: with_trailer;`

例：`http_raw_cookie: with_trailer;`

http_raw_header

検出カーソルを正規化されていないヘッダーに設定します。`http_raw_header`には、元のメッセージの変更されていないすべてのヘッダー名と値が含まれます。

`http_raw_header`ルールオプションには、パラメータの`http_raw_header.field`、`http_raw_header.request`、`http_raw_header.with_header`、`http_raw_header.with_body`、および`http_raw_header.with_trailer`が含まれます。

シンタックス：`http_raw_header: field <field_name>, <parameter>, <parameter>;`

例：`http_raw_header: field Content-Type, with_trailer;`

http_raw_header.field

指定したヘッダー名を正規化されていないHTTPヘッダーと照合します。ヘッダー名では大文字と小文字が区別されません。ヘッダー名を指定しなかった場合、HTTP インスペクタは、HTTP Cookie のヘッダー（Cookie と Set-Cookie）を除くすべてのヘッダーを確認します。

型：文字列

シンタックス：`http_raw_header: field <field_name>`

有効な値：HTTP ヘッダー名。

例：`http_raw_header: field Content-Type;`

http_raw_header.request

HTTP要求メッセージで検出されたヘッダーを照合します。応答メッセージを確認する場合は、HTTP 要求ヘッダーを使用します。`http_raw_header.request`パラメータはオプションです。

シンタックス：`http_raw_header: request;`

例：`http_raw_header: request;`

http_raw_header.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。`http_raw_header.with_header` パラメータはオプションです。

シンタックス：`http_raw_header: with_header;`

例：`http_raw_header: with_header;`

http_raw_header.with_body

ルールの別の部分で確認するのは`http_raw_header`ルールオプションではなく、HTTP メッセージの本文であることを指定します。`http_raw_header.with_body`パラメータはオプションです。

シンタックス：`http_raw_header: with_body;`

例：`http_raw_header: with_body;`

http_raw_header.with_trailer

ルールの別の部分で確認するのは`http_raw_header`ルールオプションではなく、HTTP メッセージトレーラであることを指定します。`http_raw_header.with_trailer` パラメータはオプションです。

シンタックス：`http_raw_header: with_trailer;`

例：`http_raw_header: with_trailer;`

http_raw_request

検出カーソルを正規化されていない要求行に設定します。最初のヘッダー行の特定の部分を調べるには、ルールオプションの `http_method`、`http_raw_uri`、または `http_version` のいずれかを使用します。

`http_raw_request` ルールオプションには、パラメータの `http_raw_request.with_header`、`http_raw_request.with_body`、および `http_raw_request.with_trailer` が含まれます。

シンタックス：`http_raw_request: <parameter>, <parameter>;`

例：`http_raw_request: with_header;`

http_raw_request.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。`http_raw_request.with_header` パラメータはオプションです。

シンタックス：`http_raw_request: with_header;`

例：`http_raw_request: with_header;`

http_raw_request.with_body

ルールの別の部分で確認するのは `http_raw_request` ルールオプションではなく、HTTP メッセージの本文であることを指定します。`http_raw_request.with_body` パラメータはオプションです。

シンタックス：`http_raw_request: with_body;`

例： `http_raw_request: with_body;`

http_raw_request.with_trailer

ルールの別の部分で確認するのは `http_raw_request` ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。`http_raw_request.with_trailer` パラメータはオプションです。

シンタックス：`http_raw_request: with_trailer;`

例：`http_raw_request: with_trailer;`

http_raw_status

検出カーソルを正規化されていないステータス行に設定します。ステータス行の特定の部分を調べるには、ルールオプションの `http_version`、`http_stat_code`、または `http_stat_msg` のいずれかのを使用します。

`http_raw_status` ルールオプションには、パラメータの `http_raw_status.with_body` と `http_raw_status.with_trailer` が含まれます。

シンタックス：`http_raw_status: <parameter>, <parameter>;`

例：`http_raw_status: with_body;`

http_raw_status.with_body

ルールの別の部分で確認するのは `http_raw_status` ルールオプションではなく、HTTP メッセージの本文であることを指定します。`http_raw_status.with_body` パラメータはオプションです。

シンタックス：`http_raw_status: with_body;`

例：`http_raw_status: with_body;`

http_raw_status.with_trailer

ルールの別の部分で確認するのは `http_raw_status` ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。`http_raw_status.with_trailer` パラメータはオプションです。

シンタックス：`http_raw_status: with_trailer;`

例：`http_raw_status: with_trailer;`

http_raw_trailer

検出カーソルを正規化されていない HTTP トレーラに設定します。トレーラにはメッセージの内容に関する情報が含まれています。クライアント要求で HTTP ヘッダーを作成する場合、トレーラは使用できません。

`http_raw_trailer` は、終了ヘッダーに適用されることを除いて、`http_raw_header` と同じです。HTTP のヘッダーとトレーラを検査するには、個別のルールを作成する必要があります。

http_raw_trailer ルールオプションには、パラメータの http_raw_trailer.field、http_raw_trailer.request、http_raw_trailer.with_header、http_raw_trailer.with_body が含まれます。

シンタックス：http_raw_trailer: field <field_name>, <parameter>, <parameter>;

例：http_raw_trailer: field <field_name>, request;

http_raw_trailer.field

指定したトレーラ名を正規化されていないHTTPトレーラと照合します。トレーラ名では大文字と小文字が区別されません。

型：文字列

シンタックス：http_raw_trailer: field <field_name>;

有効な値：HTTP トレーラ名。

例： http_raw_trailer: field trailer-timestamp;

http_raw_trailer.request

HTTP要求メッセージで検出されたトレーラを照合します。応答メッセージを確認する場合は、HTTP 要求トレーラを使用します。http_raw_trailer.request パラメータはオプションです。

シンタックス：http_raw_trailer: request;

例：http_raw_trailer: request;

http_raw_trailer.with_header

ルールで確認できるのは HTTP 応答ヘッダーのみであることを指定します。http_raw_trailer.with_header パラメータはオプションです。

シンタックス：http_raw_trailer: with_header;

例：http_raw_trailer: with_header;

http_raw_trailer.with_body

ルールの別の部分で確認するのは http_raw_trailer ルールオプションではなく、HTTP メッセージの本文を検査することを指定します。http_raw_trailer.with_bodyパラメータはオプションです。

シンタックス：http_raw_trailer: with_body;

例：http_raw_trailer: with_body;

http_raw_uri

検出カーソルを正規化されていない URI に設定します。

http_raw_uri ルールオプションには次のものが含まれます。

- http_raw_uri.with_header

- http_raw_uri.with_body
- http_raw_uri.with_trailer
- http_raw_uri.scheme
- http_raw_uri.host
- http_raw_uri.port
- http_raw_uri.path
- http_raw_uri.query
- http_raw_uri.fragment

シンタックス：http_raw_uri: <parameter>, <parameter>;

例：http_raw_uri: with_header, path, query;

http_raw_uri.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。http_raw_uri.with_header パラメータはオプションです。

シンタックス：http_raw_uri: with_header;

例：http_raw_uri: with_header;

http_raw_uri.with_body

ルールの別の部分で確認するのは http_raw_uri ルールオプションではなく、HTTP メッセージの本文であることを指定します。http_raw_uri.with_body パラメータはオプションです。

シンタックス：http_raw_uri: with_body;

例：http_raw_uri: with_body;

http_raw_uri.with_trailer

ルールの別の部分で確認するのは http_raw_uri ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。http_raw_uri.with_trailer パラメータはオプションです。

シンタックス：http_raw_uri: with_trailer;

例：http_raw_uri: with_trailer;

http_raw_uri.scheme

URI のスキームに対してのみ照合します。http_raw_uri.scheme パラメータはオプションです。

シンタックス：http_raw_uri: scheme;

例：http_raw_uri: scheme;

http_raw_uri.host

URI のホスト（ドメイン名）に対してのみ照合します。`http_raw_uri.host` パラメータはオプションです。

シンタックス：`http_raw_uri: host;`

例：`http_raw_uri: host;`

http_raw_uri.port

URI のポート（TCP ポート）に対してのみ照合します。`http_raw_uri.port` パラメータはオプションです。

シンタックス：`http_raw_uri: port;`

例：`http_raw_uri: port;`

http_raw_uri.path

URIのパスセクション（ディレクトリとファイル）に対してのみ照合します。`http_raw_uri.path` パラメータはオプションです。

シンタックス：`http_raw_uri: path;`

例： `http_raw_uri: path;`

http_raw_uri.query

URI のクエリパラメータに対してのみ照合します。`http_raw_uri.query` パラメータはオプションです。

シンタックス：`http_raw_uri: query;`

例：`http_raw_uri: query;`

http_raw_uri.fragment

URIのフラグメントセクションに対してのみ照合します。フラグメントは、要求したファイルの一部であり、通常はブラウザ内でのみ検出され、ネットワーク経由では送信されません。`http_raw_uri.fragment` パラメータはオプションです。

シンタックス：`http_raw_uri: fragment;`

例：`http_raw_uri: fragment;`

http_stat_code

検出カーソルをHTTP ステータスコードに設定します。HTTP ステータスコードは、100 ～ 599 の範囲の 3 桁の数字です。

`http_stat_code` ルールオプションには、パラメータの `http_stat_code.with_body` と `http_stat_code.with_trailer` が含まれます。

シンタックス：`http_stat_code: <parameter>, <parameter>;`

例：`http_stat_code: with_trailer;`

http_stat_code.with_body

ルールの別の部分で確認するのは `http_stat_code` ルールオプションではなく、HTTP メッセージの本文を検査することを指定します。`http_stat_code.with_body` パラメータはオプションです。

シンタックス：`http_stat_code: with_body;`

例：`http_stat_code: with_body;`

http_stat_code.with_trailer

ルールの別の部分で確認するのは `http_stat_code` ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。`http_stat_code.with_trailer` パラメータはオプションです。

シンタックス：`http_stat_code: with_trailer;`

例：`http_stat_code: with_trailer;`

http_stat_msg

検出カーソルを HTTP ステータスメッセージに設定します。HTTP ステータス メッセージは、HTTP ステータスコードをプレーンテキストで記述します（例：`OK`）。

`http_stat_msg` ルールオプションには、パラメータの `http_stat_msg.with_body` と `http_stat_msg.with_trailer` が含まれます。

シンタックス：`http_stat_msg: <parameter>, <parameter>;`

例：`http_stat_msg: with_body;`

http_stat_msg.with_body

ルールの別の部分で確認するのは `http_stat_msg` ルールオプションではなく、HTTP メッセージの本文であることを指定します。`http_stat_msg.with_body` パラメータはオプションです。

シンタックス：`http_stat_msg: with_body;`

例：`http_stat_msg: with_body;`

http_stat_msg.with_trailer

ルールの別の部分で確認するのは `http_stat_msg` ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。`http_stat_msg.with_trailer` パラメータはオプションです。

シンタックス：`http_stat_msg: with_trailer;`

例：`http_stat_msg: with_trailer;`

http_trailer

検出カーソルを正規化されたトレーラに設定します。トレーラにはメッセージの内容に関する情報が含まれています。クライアント要求でHTTPヘッダーを作成する場合、トレーラは使用できません。

`http_trailer` は、終了ヘッダーに適用されることを除いて、`http_header` と同じです。HTTPのヘッダーとトレーラを検査するには、個別のルールを作成する必要があります。

`http_trailer` ルールオプションには、パラメータの `http_trailer.field`、`http_trailer.request`、`http_trailer.with_header`、`http_trailer.with_body` が含まれます。

シンタックス：`http_trailer: field <field_name>, <parameter>, <parameter>;`

例：`http_trailer: field trailer-timestamp, with_body;`

http_trailer.field

指定したトレーラ名を正規化されたHTTPトレーラと照合します。トレーラ名では大文字と小文字が区別されません。

型：文字列

シンタックス：`http_trailer: field <field_name>;`

有効な値：HTTP トレーラ名。

例：`http_trailer: field trailer-timestamp;`

http_trailer.request

HTTP要求メッセージで検出されたトレーラを照合します。応答メッセージを確認する場合は、HTTP 要求トレーラを使用します。`http_trailer.request` パラメータはオプションです。

シンタックス：`http_trailer: request;`

例：`http_trailer: request;`

http_trailer.with_header

ルールの別の部分で確認するのは `http_trailer` ルールオプションではなく、HTTP メッセージのヘッダーであることを指定します。`http_trailer.with_header` パラメータはオプションです。

シンタックス：`http_trailer: with_header;`

例：`http_trailer: with_header;`

http_trailer.with_body

ルールの別の部分で確認するのは `http_trailer` ルールオプションではなく、HTTP メッセージの本文であることを指定します。`http_trailer.with_body` パラメータはオプションです。

シンタックス：`http_trailer: with_body;`

例：`http_trailer: with_body;`

http_true_ip

検出カーソルを最終的なクライアント IP アドレスに設定します。クライアントが要求を送信すると、プロキシサーバーは最終的なクライアント IP アドレスを保存します。クライアント IP アドレスは、X-Forwarded-For、True-Client-IP、またはその他のカスタム X-Forwarded-For タイプのヘッダーにリストされている最後の IP アドレスです。複数のヘッダーが存在する場合、Snort は xff_headers で定義されたヘッダーを考慮します。

http_true_ip ルールオプションには、パラメータの http_true_ip.with_header、http_true_ip.with_body、および http_true_ip.with_trailer が含まれます。

シンタックス：http_true_ip: <parameter>, <parameter>;

例：http_true_ip: with_header;

http_true_ip.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。http_true_ip.with_header パラメータはオプションです。

シンタックス：http_true_ip: with_header;

例：http_true_ip: with_header;

http_true_ip.with_body

ルールの別の部分で確認するのは http_true_ip ルールオプションではなく、HTTP メッセージの本文であることを指定します。http_true_ip.with_body パラメータはオプションです。

シンタックス：http_true_ip: with_body;

例：http_true_ip: with_body;

http_true_ip.with_trailer

ルールの別の部分で確認するのは http_true_ip ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。http_true_ip.with_trailer パラメータはオプションです。

シンタックス：http_true_ip: with_trailer;

例： http_true_ip: with_trailer;

http_uri

検出カーソルを正規化された URI バッファに設定します。

- http_uri.with_header
- http_uri.with_body
- http_uri.with_trailer
- http_uri.scheme
- http_uri.host

- http_uri.port
- http_uri.path
- http_uri.query
- http_uri.fragment

シンタックス：http_uri: <parameter>, <parameter>;

例：http_uri: with_trailer, path, query;

http_uri.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。http_uri.with_header パラメータはオプションです。

シンタックス：http_uri: with_header;

例：http_uri: with_header;

http_uri.with_body

ルールの別の部分で確認するのは http_uri ルールオプションではなく、HTTP メッセージの本文であることを指定します。http_uri.with_body パラメータはオプションです。

シンタックス：http_uri: with_body;

例：http_uri: with_body;

http_uri.with_trailer

ルールの別の部分で確認するのは http_uri ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。http_uri.with_trailer パラメータはオプションです。

シンタックス：http_uri: with_trailer;

例：http_uri: with_trailer;

http_uri.scheme

URI のスキームに対してのみ照合します。http_uri.scheme パラメータはオプションです。

シンタックス：http_uri: scheme;

例：http_uri: scheme;

http_uri.host

URI のホスト（ドメイン名）に対してのみ照合します。http_uri.host パラメータはオプションです。

シンタックス：http_uri: host;

例：http_uri: host;

http_uri.port

URI のポート（TCP ポート）に対してのみ照合します。http_uri.port パラメータはオプションです。

シンタックス：http_uri: port;

例：http_uri: port;

http_uri.path

URI のパス（ディレクトリとファイル）に対してのみ照合します。http_uri.path パラメータはオプションです。

シンタックス：http_uri: path;

例：http_uri: path;

http_uri.query

URI のクエリパラメータに対してのみ照合します。http_uri.query パラメータはオプションです。

シンタックス：http_uri: uri;

例：http_uri: query;

http_uri.fragment

URI のフラグメントセクションに対してのみ照合します。フラグメントは、要求したファイルの一部であり、通常はブラウザ内でのみ検出され、ネットワーク経由では送信されません。http_uri.fragment パラメータはオプションです。

シンタックス：http_uri: fragment;

例：http_uri: fragment;

http_version

検出カーソルを HTTP バージョンバッファの先頭に設定します。http_version は、さまざまな HTTP バージョンを受け入れます。最も一般的に見られるバージョンは HTTP/1.0 と HTTP/1.1 です。http_version ルールオプションには、パラメータの http_version.request、http_version.with_header、http_version.with_body、および http_version.with_trailer が含まれます。

シンタックス：http_version: <parameter>, <parameter>;

例：http_version; content:"HTTP/1.1";

http_version.request

HTTP 要求で検出されたバージョンを照合します。応答メッセージを確認する場合は、要求バージョンを使用します。http_version.request パラメータはオプションです。

シンタックス：http_version: request;

例：`http_version: request;`

http_version.with_header

ルールで確認できるのは HTTP メッセージのヘッダーのみであることを指定します。`http_version.with_header` パラメータはオプションです。

シンタックス：`http_version: with_header;`

例：`http_version: with_header;`

http_version.with_body

ルールの別の部分で確認するのは `http_version` ルールオプションではなく、HTTP メッセージの本文であることを指定します。`http_version.with_body` パラメータはオプションです。

シンタックス：`http_version: with_body;`

例：`http_version: with_body;`

http_version.with_trailer

ルールの別の部分で確認するのは `http_version` ルールオプションではなく、HTTP メッセージのトレーラであることを指定します。`http_version.with_trailer` パラメータはオプションです。

シンタックス：`http_version: with_trailer;`

例：`http_version: with_trailer;`

http_version_match

標準の HTTP バージョンと照合する HTTP バージョンのリストを指定します。複数のバージョンは余白文字で区切ります。HTTP 要求またはステータス行には、バージョンが含まれる場合があります。そのバージョンが存在する場合、Snort はこのバージョンを `http_version_match` で指定されたリストと比較します。

そのバージョンの形式の `[0-9].[0-9]` がない場合は、不正な形式と見なされます。`1.0` または `1.1` ではない `[0-9].[0-9]` の形式のバージョンは、その他と見なされます。

型：文字列

シンタックス：`http_version_match: <version_list>`

有効な値：`1.0`、`1.1`、`2.0`、`0.9`、`other`、`malformed`

例：`http_version_match: "1.0 1.1";`

js_data

検出カーソルを正規化された JavaScript データに設定します。このオプションは、拡張 JavaScript ノーマライザに固有です。

シンタックス：`js_data;`

例：`js_data;`

vba_data

検出カーソルを Microsoft Office Visual Basic for Applications マクロバッファに設定します。

シンタックス：`vba_data;`

例：`vba_data;`

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。