Snort 3について：ステートフルシグニチャ評価Byte_Jump

内容

はじめに

このドキュメントでは、7.4以降でSnort 3に追加された新しいテクニックについて説明します。

背景説明

- Snort 3検出モジュールは、ブロックモードで動作します。このアプローチは、パフォーマンスの優位性と実装の簡素化（比較的）を実現しますが、複数のデータブロックにまたがるシグニチャの検出には若干の制限があります。
- ユーザエクスペリエンスを向上させるために、Snortには次のような改善機能がすでに実装されています。
  1. フロービットを使用すると、ルールライターはネットワークフローにユーザ定義プロパティをマークできます。このプロパティは、フローからの任意のパケットで設定、クリア、およびテストできます（これは、パケット上のいくつかの大きなシグニチャについて結論を出す方法です）。
- ストリームモジュールは、ワイヤパケットを再構成されたパケットに蓄積します。これは、生のパケットよりも大きく、意味のあるブロックです。再構成されたパケットに対してIPSルールを評価することで、全体像を把握し、より大きなパターン（シグニチャ）に一致

する機会が増えます。
- 再構成されたパケットが新しいデータを提示するだけでなく、検出によってすでに処理された以前のデータの一部を含む場合もあります。この場合も、蓄積されたデータのブロックによって、フロー上で（ある程度）後方に広がるシグニチャを検出できます。
- ストリームスプリッタはフローをブロックに分割しますが、切断点は攻撃者がパターン検出を回避するために使用できる脆弱なポイントである可能性があります。そのため、Snortには、分割を予測不可能にするためのジッタ機構が実装されています。これにより、攻撃者の分析がさらに複雑になります。

最新情報

ステートフルシグニチャ評価は、リストに追加できる新しい手法です。複数のブロックに対するIPSルール評価を有効にすることで、検出機能を拡張します。したがって、現在のブロックにデータがない場合、ルールは即座にミスマッチせず、代わりに、より多くのデータが到着するのを待ちます。

対応プラットフォーム

最低限のソフトウェアおよびハードウェアプラットフォーム

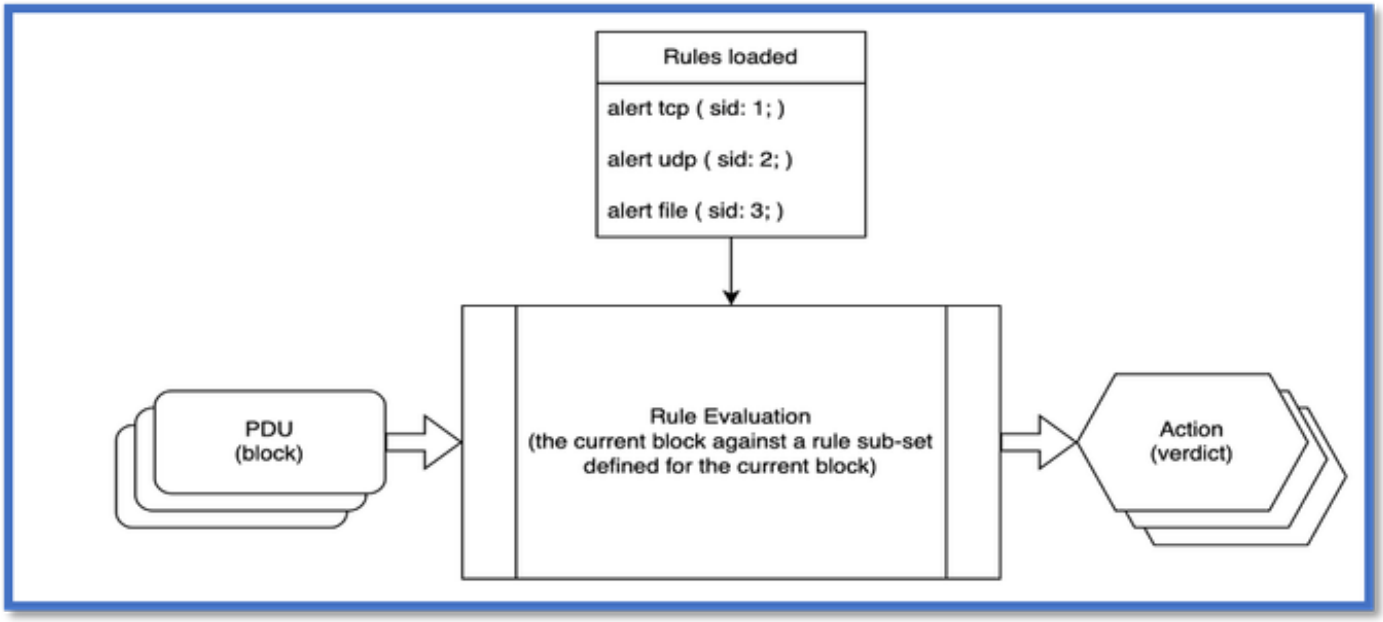
サポートされるManagerの最小バージョン	管理対象デバイス	サポートされる管理対象デバイスの最小バージョンが必要	注意事項
Management Center 7.4.0	FTD	7.4.0	Snort 3のみ
デバイスマネージャ 7.4.0	FDM管理をサポートするすべてのFTD	7.4.0	Snort 3のみ

機能の詳細

機能の説明

この仕組みを説明しましょう

検出モジュールのワークフローが図に示されています。トラフィック処理段階で、モジュールにはすでにロードされているすべてのルールがあり、モジュールは1つずつデータブロックを受け入れ、ルールを評価し、プロセスステートフルシグニチャ評価ブロックに対して実行されるアクションを定義します。



スキームに関する注意：

1. 現在のデータ・ブロックにルール・サブセットが定義されると、ルール・サブセット内の各ルールは、他のルールとは別に評価されます。
2. 各データブロックは、他のブロックとは独立して評価されます。
3. データブロックは、現在のパケットに対して評価されるIPSバッファセットの抽象化です。
4. アクションは、現在のパケットに対して評価されたアクションのリストです。最終的な判定は後で行われます。

ステートフルシグニチャ評価の仕組みを理解するには、一般的なIPSルールの評価方法と、データブロックがストリームを形成する方法を確認します。

共通ルール評価

IPSルールは次の形式で表示できます。

```
action protocol source → destination ( option_1: parameters; option_2: parameters;
option_3: parameters; gid: 1; sid: 1; meta_option_1; meta_option_2; meta_option_3; )
```

場所：

action：ルールが起動した場合のパケットのIPSアクション

protocol：照合するプロトコル

送信元、宛先:IPアドレスとポート

option_1、option_2、option_3：ルール評価の一部であるIPSオプション

gid、sid：ルールを識別する一意のペア（メタデータ・オプションと同様）

meta_option_1、meta_option_2、meta_option3：ルールのメタデータ（メッセージ、クラス・タイプ、参照など）。これらのオプションは、ルールの評価には関与しません。

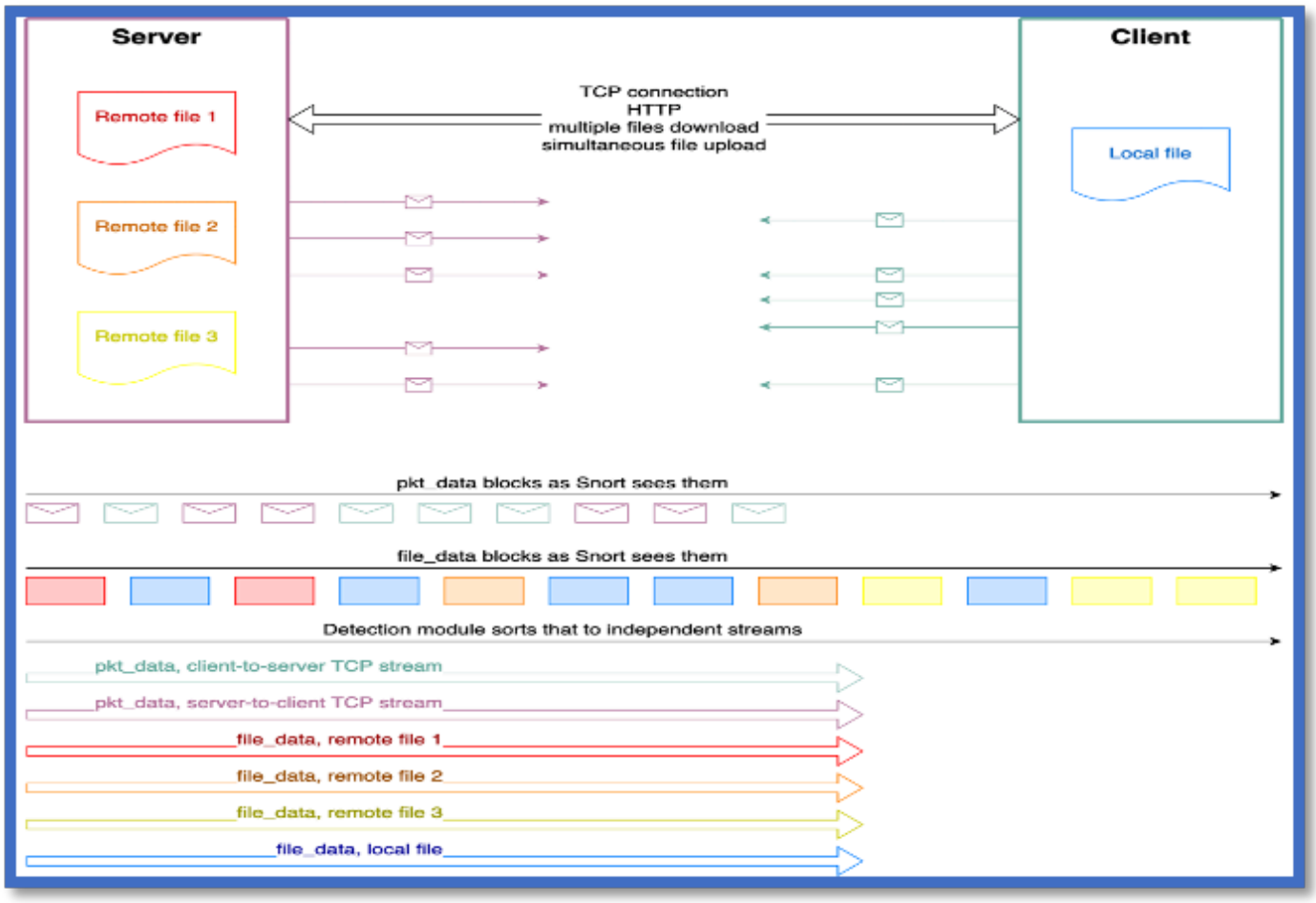
- プロトコル、送信元、および宛先がルールヘッダーを形成します。これは、ネットワークフロー（評価のために受け入れるフロー）のフィルタのように機能します。括弧内はすべて規則本文です。ルール本体のIPSオプション（ルールメタデータを除く）は、データブロックに対して評価されるオプションです。次の条件に準拠しています。
- オプションは、左から右の順に厳密に評価されます。
    1. 次の2つの主要なタイプのいずれかになります。
    2. buffer setterを使用すると、現在のパケットのIPSバッファが選択されます。
- その他（パターン検索、数値演算、カーソル操作、flowbit演算）
- カーソルは、選択したIPSバッファ内の位置を追跡するために使用されます。

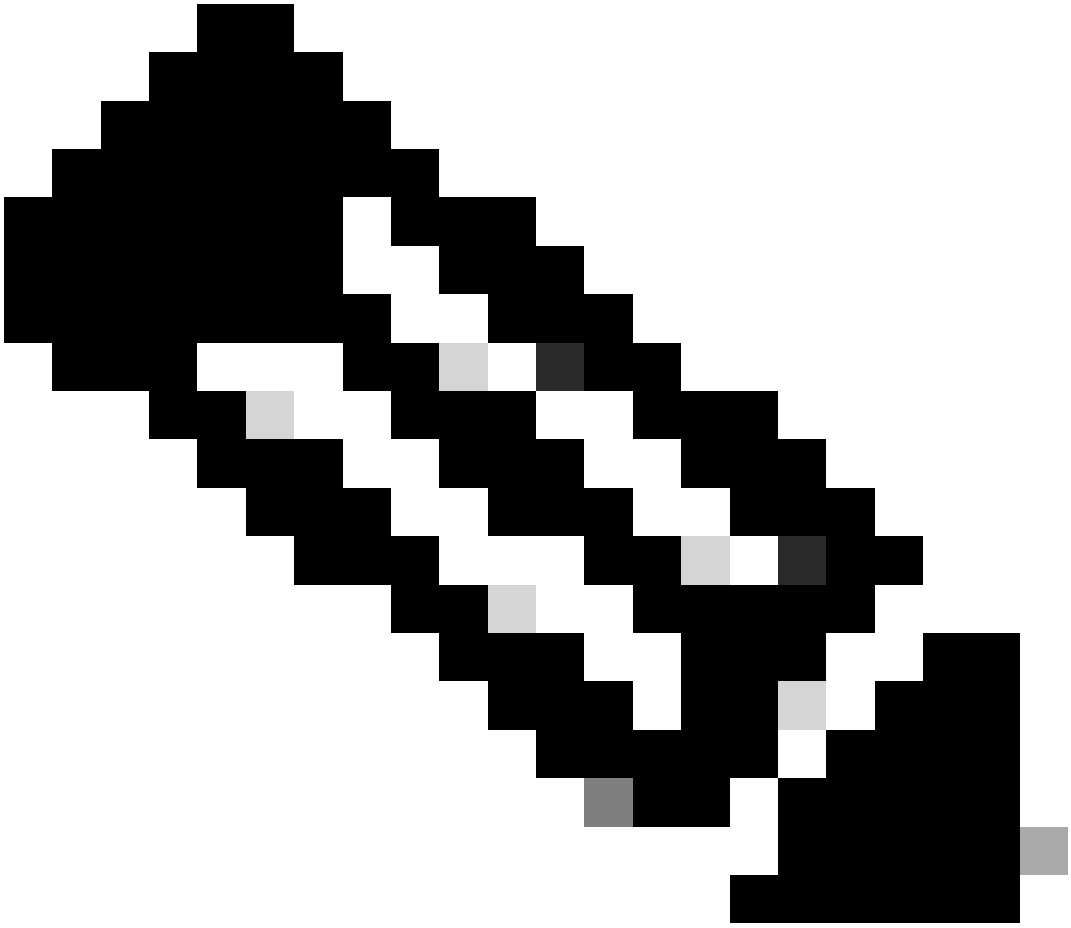
- オプションは次のいずれかになります。
    1. 'absolute'。カーソル位置に依存しないことを意味します。
    2. 'relative'は、カーソル位置から評価を開始することを意味します。
- オプションが選択されたIPSバッファからカーソルを設定しようとすると、失敗し、ルール全体が不一致になります（データ不足のため）
- 最後の点は、検出モジュールの制限です。Snortのリソースが無制限の場合、データが使用可能になると（より多くの有線パケットが着信すると）、ルールを評価するために見つかったすべてのデータが何度もキャッシュされます。

データストリームとIPSバッファ

- データストリームは、同じ送信元からの連続した形式のバイトのストリームです。これは、ステートフルな評価をサポートするために提示された新しい概念です。ブロック間のルール評価は、同じ論理データ（ファイル、純粋なTCPストリーム、JavaScriptテキストなど）内で行う必要があります。

- 一般に、検出モジュールが受信するデータブロックは次のようになります。
    - 別のIPSバッファからコピーする（たとえば、pkt_dataとfile_dataは同じではありません）
    - 別のストリームに属する
    - ストリームを形成しない（rawパケットから生成されるバッファ）
    - 連続ストリームを形成しない(ICMP、UDP)
    - 順不同（HTTP部分応答）
    - 繰り返されるデータを含む（http_inspect.script_detectionやHTTP Chunked Responseなどの累積ブロック）
- 検出モジュールは、同じストリームのブロックのみを連結するように状況をソートできます。ソートしないと、評価プロセスは、ブロックをインターリーブすることによって不要な干渉を検出します。


Server
Remote file 1
Remote file 2
Remote file 3
TCP connection
HTTP
multiple files download
simultaneous file upload
Client
Local file
pkt_data blocks as Snort sees them
file_data blocks as Snort sees them
Detection module sorts that to independent streams
pkt_data, client-to-server TCP stream
pkt_data, server-to-client TCP stream
file_data, remote file 1
file_data, remote file 2
file_data, remote file 3
file_data, local file



注：次の例は、HTTPクライアントが複数のファイルを同時にアップロードおよびダウンロードするケースを示しています。

- 現在、ストリームを表すことができるのは、pkt_dataとfile_dataの2つのIPSバッファだけです。ここで、
  1. pkt_dataは、TCPプロトコルの2つのストリームを形成します（クライアントからサーバへの方向とサーバからクライアントへの方向）
  2. file_dataは、ファイル、MIME添付ファイル、およびその他のプロトコルデータ（HTTP HTMLページやその他のコンテンツタイプなど）のストリームを形成する必要があります
- ステートフル評価は、データストリーム内で厳密に実行されます。

ルールの継続

- 前述のセクションは、現在のIPSバッファからカーソルを外した場合にIPSオプションが不一致であるという記述で終了します。ただし、IPSバッファがデータストリームを形成する場合、ステートフルシグニチャ評価機能によって、Snortフローオブジェクトにルール評価

コンテキストが介入および保存されます。保存された評価コンテキスト（状態）は、ルールの継続と呼ばれます。ステートフルシグニチャ評価は、より多くのデータが利用可能になるまで、ルールの最終的な判定を延期します。
- ルールの継続には、IPSバッファ名、バッファソース、およびターゲットカーソル位置の3つの主要部分があります（バッファソースはデータストリームの固有識別子です）。
- 検出モジュールによってデータブロックが処理されると、後続のアクションが実行されます。
    - ステートフルシグニチャ評価は、次の場合にルールの継続を作成し、フローに付加します。
        - IPSオプション（byte_jump、content、pcreなど、カーソル位置を更新するもの）は、現在のIPSバッファの後ろにカーソルを設定する
        - 現在のIPSバッファはデータストリームをサポートします。
        - 現在のIPSバッファは、この時点でデータストリームを形成します。
- ステートフルシグニチャ評価では、次の場合に、作成したばかりのルールの継続が取り消され、フローから削除されます。
    - IPSルールが現在のデータブロックで起動しました（ルールはブロックの他の場所で一致します）
- ステートフルシグニチャ評価は、保留中のルール継続を拒否し、次の場合にはフローから削除します。
    - IPSバッファは連続ストリームを形成しません(たとえば、ブロックに繰り返しデータがあるか、ギャップがある（データの一部が失われた、またはブロックが正常に動作していない）。
- ステートフルシグニチャ評価は、次の場合に新しいデータを使用してターゲットカーソル位置を更新します。
    - ルールの継続からのバッファソースは、選択したバッファソースと同じです
    - IPSバッファが連続ストリームを形成
- ステートフルシグニチャ評価は、次の場合にIPSルールエンジンにルール継続を送信します。
    1. 選択したIPSバッファ内の目的のカーソル位置（つまり、最終的にルールの評価を完了するために必要なすべてのデータを受信しました）。

ユーザ設定

- ルールの連続にはメモリが必要なため、Snortでは連続したルールを無制限に保存することはできません。制限を制御するための設定オプションがあります。
    1. Detection.max_continuations_per_flow = 1024：フロー{ 0:65535 }に同時に格納された連続の最大数
- ステートフルシグニチャ評価が限界に達すると、最も古いルールの続きを新しいルールに置き換えます。
- フローに存在する最も古いルールの継続が長すぎるので、ルール評価を再開する条件を満たしていません。
- さらに、IPSルールを微調整するために使用できるペグカウントが多数あります（メインの焦点である必要があります）。また、制限（必要に応じて）もあります。
    1. detection.cont_creations：作成された連続の合計数（合計）
    2. detection.cont_recalls：リコールされた継続回数の合計(sum)
    3. detection.cont_flows：継続を使用したフローの総数(sum)

4. detection.cont_evals：条件を満たした連続の合計数（合計）
5. detection.cont_matches：一致した連続の総数（合計）
6. detection.cont_mismatches：不一致の連続の総数(sum)
7. detection.cont_max_num：フローごとの同時継続のピーク数（最大）
8. detection.cont_match_distance：一致した連続によってジャンプされた合計バイト数(sum)
9. detection.cont_mismatch_distance：連続の不一致によってジャンプされたバイト数の合計(sum)

トラブルシューティング

この機能は既存の検出プロセスの拡張機能であるため、では明示的なトラブルシューティングを行うことはできません。検出で障害が発生した場合は、ルール、設定、またはトラフィックを調べる必要があります。

問題例

問題：説明

- たとえば、署名がファイルの先頭と末尾を同時にチェックする必要があるとします。
- たとえば、この構造のターゲットファイル（ヘッダー、ボディ、メタデータ）では、メタデータのいずれかに0の値があるかどうかを確認する必要があります。

- ファイルバイト： e1 f3 22 03 7f ff xx ... xx 01 00 02 00
    - e1 f3 22 03：ファイルタイプを識別するマジックナンバーの4バイト
    - 7f ff - 2バイト（本体サイズ）
    - xx xx ... xx – 一部のデータの32 kb
    - 01 00 02 00：タグ値形式の4バイトのメタデータ（各1バイト）
- IPSルールは次のようになります。 alert file (file_data; content:"|e1f32203|",fast_pattern; byte_jump:2,0,relative; content:"00",within:4, relative; sid: 1;)
    - どこから?
        - ファイルプロトコルにより、ルールは再構築されたパケットのみを受け入れる（rawパケットはステートフルシグニチャ評価に参加しない）
        - 'file_dataオプションは、ストリームを形成できるファイルデータバッファを選択します
        - 1番目のコンテンツオプションは高速パターンであり、マジック番号をチェックします（それが目的のファイルタイプである場合）
        - byte_jumpオプションはファイル本体のサイズを読み込み、ファイル本体を飛び越える
        - 2番目のコンテンツオプションは、パラメータ制限検索の深さの範囲内でメタデータ値の最終チェックを実行し、オプションを相対的にします。

問題：ソリューション

ルールは次のように評価されます。

ファイルヘッダーと本文の一部を含む（8kBサイズの）最初のパケットについて：

1. IPSバッファfile_dataが選択されます。カーソルは0番目のバイトe1を指しています。
2. fast patternオプションは、マジックナンバーの直後のバイト7fを指すカーソル位置に一致し、設定します。
3. byte_jumpオプションは、ファイル本体のサイズを2バイト読み込みます。この2バイトでカーソルが更新されます。byte_jumpは32768バイト以上のジャンプを計算します。
4. ステートフルシグニチャ評価では、24578バイトを超えるルールの継続が必要になります(32768 -（8kB - 4バイトのヘッダー – 2バイトの本文サイズ））。
5. byte_jumpオプションではカーソル位置をそこまで設定できないため、ルール全体が一致しません。

2番目のパケット（16kBサイズ）で、ファイル本体部分を伝送する場合：

1. ステートフルシグニチャ評価では、保留中のルールの継続を確認します。
2. バッファは名前で選択され、file_dataが使用可能であること、および新しいデータサイズが16384であることを確認します。
3. 更新されたカーソルは、8194バイトがまだ必要であることを示します(24578 - 16384)
4. ルールは再開されません。

3番目のパケット（サイズ8198）では、ファイル本体の部分とメタデータが伝送されます。

1. ステートフルシグニチャ評価では、保留中のルールの継続を確認します。
2. バッファは名前で選択され、file_dataが使用可能で、新しいデータサイズが8198であることがわかります。
3. 更新されたカーソルは、バッファに十分なデータがあることを示します。カーソル位置は8194です。
4. ステートフルシグニチャ評価は、ルールの継続を削除します。
5. ステートフルシグニチャ評価は、カーソルがバイト01を指している2番目のコンテンツオプションからルール評価を再開します。
6. contentオプションは、検索された2番目のバイトで一致を見つけます。
7. ルール全体が最終的に起動します。

制限事項の詳細と一般的な問題

制限事項およびその他の考慮事項

- ステートフルシグニチャ評価の実装により、Snortは設定をリロードするときに、保留中のルールの継続をすべてドロップします。ドロップされてもルールが継続すると、次のデータブロックが検出モジュールに送信されるまでSnortメモリが占有されることに注意してください。
- ステートフル評価におけるIPSルールのルール遅延機能は、共通ルール評価の場合と同様に動作します。異なるデータブロック上のルール部分の評価時間を要約する。時間が制限を超えると、ルール評価は短絡を行い、早く終了します。
- Flowbits操作はその意味を保持しますが、それでも「静的」オプションと同様に動作します

。
フロービットセット/クリア/テスト操作は、現在既知のコンテキスト内で実行されます。したがって、flowbitオプションがルールの継続で評価される場合、ルールが評価を開始した時点ではなく、現在の環境（flowbitsセット）が考慮されます。

また、ルールライターはファストパターンの場所に注意を払う必要があります。

規則の任意の部分に含まれる可能性がある場合でも、fast-patternオプションは規則全体の前に評価されます。ルール評価がトリガーされます。ステートフルシグニチャ評価ベースのルールの場合は、ルール継続ポイントがfast-patternオプションの後にある必要があります。
さらに、IPSルールの評価では、複数のルールを連続して実行できます（同時に実行されるのではなく、次々に実行されます）。ルール本文の任意のオプションに継続を指定できるため、ルールライターは同じIPSルールを使用してデータストリームの異なる場所で追加チェックを実行できます。

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。