カスタム侵入ルール

以下のトピックでは、侵入ルール エディタの使用方法について説明します。

カスタム侵入ルールの概要

侵入ルールは、ネットワークの脆弱性を不正利用する試みを検出するために使用するキーワードや引数です。ネットワーク トラフィックの分析では、パケットを各ルールで指定した条件と比較します。パケットのデータがルールで指定したすべての条件に一致すると、そのルールがトリガーされます。アラート ルールであれば、侵入イベントが生成されます。通過ルールであれば、トラフィックを無視します。インライン展開の廃棄ルールでは、システムがパケットを破棄してイベントを生成します。侵入イベントは、Secure Firewall Management Center の Web インターフェイスから表示して評価できます。

システムの侵入ルールには、共有オブジェクト ルールと標準テキスト ルールの 2 種類があります。Talos インテリジェンス グループ では、共有オブジェクト ルールを使うことにより、従来の標準テキスト ルールではできなかった方法で脆弱性に対する攻撃を検出できます。共有オブジェクト ルールを作成することはできません。独自の侵入ルールを作成する場合は、標準テキスト ルールを作成します。

発生する可能性のあるイベントのタイプを調整するために、カスタム標準テキスト ルールを作成することができます。このマニュアルでは特定のエクスプロイトの検出を目的とするルールについて説明することもありますが、優秀なルールのほとんどは、特定の既知のエクスプロイトではなく既知の脆弱性を悪用しようとするトラフィックをターゲットとすることに注意してください。ルールを作成してルールのイベント メッセージを指定することにより、攻撃とポリシー回避を示唆するトラフィックをより簡単に識別できます。

カスタム侵入ポリシーでカスタム標準テキストルールを有効にすると、一部のルールキーワードと引数では、トラフィックを特定の方法で最初に復号または前処理する必要があることに留意してください。この章では、前処理を制御するネットワーク分析ポリシーで設定する必要があるオプションについて説明します。注意点として、必要なプリプロセッサを無効にすると、システムは自動的に現在の設定でプリプロセッサを使用します。ただし、ネットワーク分析ポリシーの Web インターフェイスではプリプロセッサは無効のままになります。

⚠

注意 作成した侵入ルールを実稼働環境で使用する前に、制御されたネットワーク環境で必ずテストしてください。不適切に作成された侵入ルールは、システムのパフォーマンスに重大な影響を与える可能性があります。

侵入ルールエディタのライセンス要件

Threat Defense ライセンス

IPS

従来のライセンス

保護

侵入ルールエディタの要件と前提条件

モデルのサポート

任意

サポートされるドメイン

任意

ユーザの役割

- 管理者
- 侵入管理者

ルールの詳細

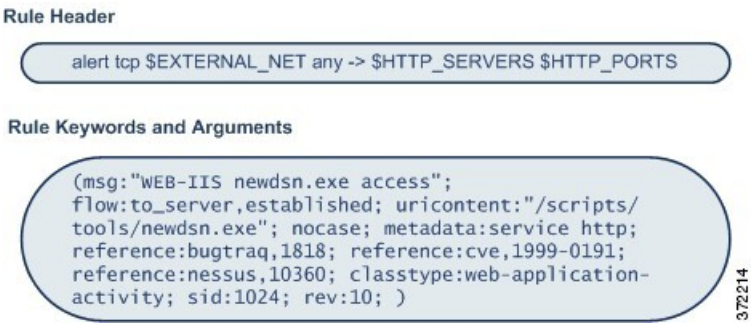
すべての標準テキスト ルールには、ルール ヘッダーとルール オプションという 2 つの論理セクションが含まれています。ルール ヘッダーの内容は次のとおりです。

- ルールのアクションまたはタイプ
- プロトコル
- 送信元および宛先の IP アドレスとネットマスク
- 送信元から宛先へのトラフィック フローを示す方向インジケータ
- 送信元ポートと宛先ポート

ルール オプション セクションの内容は次のとおりです。

- イベント メッセージ
- キーワードとそのパラメータおよび引数
- ルールをトリガーとして使用するためにパケットのペイロードが一致する必要があるパターン
- パケットのどの部分をルール エンジンで検査するかの指定

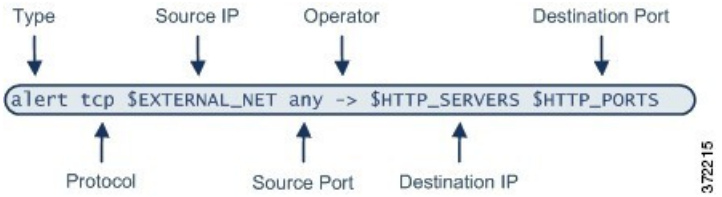
次の図に、ルールの構成要素を示します。



ルールのオプションセクションは、カッコで囲まれたセクションであることに注意してください。侵入ルール エディタは、標準テキスト ルールの作成を支援する使いやすいインターフェイスを備えています。

侵入ルール ヘッダー

すべての標準テキストルールおよび共有オブジェクトルールに、パラメータと引数を含むルール ヘッダーがあります。ルール ヘッダーの構成要素を以下に示します。



次の表では、上記のルール ヘッダーの各部分について説明します。

表 *1:* ルール ヘッダーの値

ルールヘッダーのコンポーネント	値の例	機能
操作	`alert`	トリガー時に侵入イベントを生成します。
プロトコル	`tcp`	TCP トラフィックのみをテストします。
送信元 IP アドレス	`$EXTERNAL_NET`	内部ネットワーク上に存在しないホストから送られてきたトラフィックをテストします。
送信元ポート	`any`	発信元ホスト上の任意のポートから送られてきたトラフィックをテストします。
演算子	`->`	（このネットワーク上の Web サーバーに向かう）外部トラフィックをテストします。
宛先 IP アドレス	`$HTTP_SERVERS`	この内部ネットワーク上の Web サーバとして指定された任意のホストに送られるトラフィックをテストします。
宛先ポート	`$HTTP_PORTS`	この内部ネットワーク上の HTTP ポートに送られるトラフィックをテストします。

（注） 前述の例では、ほとんどの侵入ルールの場合と同様に、デフォルト変数が使用されています。

関連トピック

変数セット

侵入ルール ヘッダー アクション

各ルールヘッダーには、パケットがルールをトリガーとして使用したときにシステムで行われるアクションを指定するパラメータが 1 つ含まれています。アクションが *alert* に設定されたルールは、それをトリガーとして使用したパケットに対する侵入イベントを生成し、そのパケットの詳細をログに記録します。アクションが *pass* に設定されたルールは、それをトリガーとして使用したパケットに関するイベントを生成せず、そのパケットの詳細も記録しません。

（注） インライン展開において、ルール状態が [ドロップしてイベントを生成する（Drop and Generate Events）] に設定されたルールは、それをトリガーとして使用したパケットに対する侵入イベントを生成します。また、パッシブ展開で廃棄ルールを適用した場合は、ルールがアラートルールとして機能します。

デフォルトでは、パス ルールがアラート ルールをオーバーライドします。パス ルールを作成することで、アラート ルールを無効にする代わりに、パス ルールで定義された基準を満たすパケットが特定の状況でアラートルールをトリガーとして使用しないことを指定できます。たとえば、ユーザ "anonymous" として FTP サーバにログインする試行を検索するルールをアクティブのままにする必要があるとします。ただし、1 つ以上の正式な匿名 FTP サーバがネットワークに存在する場合、そのような特定のサーバで匿名ユーザにより最初のルールがトリガーとして使用されないことを指定するパス ルールを作成し、アクティブにすることができます。

侵入ルール エディタで、[アクション（Action）] リストからルール タイプを選択します。

侵入ルール ヘッダー プロトコル

各ルール ヘッダーで、ルールにより検査されるトラフィックのプロトコルを指定する必要があります。次のネットワーク プロトコルを分析対象として指定できます。

- ICMP（Internet Control Message Protocol）
- インターネット プロトコル（IP）

（注） プロトコルが `ip` に設定されている場合、システムは侵入ルール ヘッダー内のポート定義を無視します。

- 伝送制御プロトコル（TCP）
- ユーザー データグラム プロトコル（UDP）

TCP、UDP、ICMP、IGMP など、IANA によって割り当てられたすべてのプロトコルを検査するには、プロトコル タイプとして **IP** を使用します。

（注） 現在のところ、IP ペイロード内の次のヘッダー（TCP ヘッダーなど）でパターンを照合するルールを作成することはできません。代わりに、最後にデコードされたプロトコルからコンテンツ照合が始まります。次善策として、ルール オプションを使用して TCP ヘッダー内のパターンを照合できます。

侵入ルール エディタで、[プロトコル（Protocol）] リストからプロトコル タイプを選択します。

関連トピック

侵入ルール ヘッダー プロトコル （5 ページ）

侵入ルール ヘッダーの方向

ルールによる検査対象となるパケットが進むべき方向を、ルール ヘッダー内で指定できます。以下の表は、それらのオプションを示しています。

表 *2:* ルール ヘッダー内の方向オプション

使用するフィルタ	テスト対象
指向性	指定された送信元 IP アドレスから指定された宛先 IP アドレスに向かうトラフィックのみ
双方向	指定された送信元 IP アドレスと宛先 IP アドレスの間を移動するすべてのトラフィック

侵入ルール ヘッダーの送信元と宛先の IP アドレス

パケット検査の対象を、特定の IP アドレスから発信されたパケットまたは特定の IP アドレスに向かうパケットに制限すると、システムが実行しなければならないパケット検査の量が減ります。さらに、ルールをより具体化し、送信元および宛先 IP アドレスが疑わしい動作を示していないパケットに対してルールがトリガーとして使用される可能性をなくすと、誤検出も減ります。

ヒント システムは IP アドレスのみを認識し、送信元/宛先 IP アドレスのホスト名を受け入れません。

侵入ルール エディタの [送信元 IP（Source IPs）] フィールドと [宛先 IP（Destination IPs）] フィールドで、送信元および宛先の IP アドレスを指定します。

標準テキスト ルールの作成時には、必要に応じて、さまざまな方法で IPv4 アドレスと IPv6 アドレスを指定できます。単一の IP アドレス、`any`、IP アドレス リスト、CIDR 表記、プレフィクス長、またはネットワーク変数を指定できます。加えて、1 つの特定の IP アドレスまたは IP アドレスのセットを除外するよう指定できます。IPv6 アドレスを指定するときには、RFC 4291 で定義された任意のアドレス指定規則を使用できます。

侵入ルールの IP アドレスの構文

次の表では、送信元と宛先の IP アドレスを指定するさまざまな方法を要約します。

表 *3:* 送信元/宛先 *IP* アドレスの構文

指定する項目	使用するフィルタ	例
任意の IP アドレス	任意	任意
1 つの特定の IP アドレス	IP アドレス 同じルール内に IPv4 と IPv6 の送信元アドレスと宛先アドレスを混在させないでください。	`192.168.1.1` `2001:db8::abcd`
IP アドレスのリスト	複数の IP アドレスをカンマで区切り、それを大カッコ（`[]`）で囲む	`[192.168.1.1,192.168.1.15]` `[2001:db8::b3ff, 2001:db8::0202]`

指定する項目	使用するフィルタ	例
IP アドレスのブロック	IPv4 CIDR ブロックまたは IPv6 アドレス プレフィクス表記	`192.168.1.0/24` `2001:db8::/32`
特定の 1 つの IP アドレスまたはアドレス セットを除くすべて	拒否する IP アドレスの前に付ける「`!`」記号	`!192.168.1.15` `!2001:db8::0202:b3ff:fe1e`
特定の 1 つ以上の IP アドレスを除く、IP アドレスブロック内のすべて	アドレス ブロックの後に、除外アドレスのリストまたはブロック	`[10.0.0/8, !10.2.3.4, !10.1.0.0/16]` `[2001:db8::/32, !2001:db8::8329, !2001:db8::0202]`
ネットワーク変数で定義された IP アドレス	`$` で始まる大文字の変数名 プリプロセッサルールは、侵入ルールで使われているネットワーク変数で定義されたホストとは無関係に、イベントをトリガーできることに注意してください。	`$HOME_NET`
IP アドレス変数で定義されたアドレスを除く、すべての IP アドレス	大文字の変数名の前に `!$` を付ける	`!$HOME_NET`

以下の説明では、いくつかの IP アドレス入力方法に関する追加情報を提供します。

任意の IP アドレス

任意の IPv4 または IPv6 アドレスを示す「`any`」という単語を、ルールの送信元 IP アドレスまたは宛先 IP アドレスとして指定できます。

たとえば、次のルールでは [Source IPs] フィールドと [Destination IPs] フィールドで引数 **any** を使用して、任意の IPv4 または IPv6 の送信元または宛先アドレスを持つパケットを評価します。

```
alert tcp any any -> any any
```

また、任意の IPv6 アドレスを示すために `::` を指定することもできます。

複数の IP アドレス

次の例に示すように、カンマを使って複数の IP アドレスを区切り、オプションで、非拒否リストを大カッコで囲むことにより、個別の IP アドレスを列挙できます。

```
[192.168.1.100,192.168.1.103,192.168.1.105]
```

IPv4 アドレスと IPv6 アドレスのいずれかだけを列挙することも、任意に組み合わせて列挙することもできます（次の例を参照）。

```
[192.168.1.100,2001:db8::1234,192.168.1.105]
```

以前のソフトウェア リリースでは IP アドレス リストを大カッコで囲む必要がありましたが、現在ではこれが必須でないことに注意してください。また、オプションで、リストを入力するときに各カンマの前または後にスペースを含めることができます。

（注） 否定リストは、大カッコで囲む必要があります。

また、IPv4 クラスレス ドメイン間ルーティング（CIDR）表記または IPv6 プレフィクス長を使用してアドレス ブロックを指定することもできます。次に例を示します。

- 192.168.1.0/24 は、サブネット マスク 255.255.255.0 の 192.168.1.0 ネットワーク内の IPv4 アドレス、つまり 192.168.1.0 ～ 192.168.1.255 を指定します。
- 2001:db8::/32 は、プレフィクス長 32 ビットの 2001:db8:: ネットワーク内の IPv6 アドレス、つまり 2001:db8:: ～ 2001:db8:ffff:ffff:ffff:ffff:ffff:ffff を指定します。

ヒント IP アドレスのブロックを指定する必要があるが、CIDR またはプレフィクス長表記を単独で使ってそれを表現できない場合は、1 つの IP アドレス リスト内でいくつかの CIDR ブロックとプレフィクス長を使用できます。

IP アドレスの否定

特定の IP アドレスを否定するために感嘆符（!）を使用できます。つまり、1 つ以上の特定の IP アドレスを除く、すべての IP アドレスに一致させることができます。たとえば、`!192.168.1.1` は 192.168.1.1 以外の任意の IP アドレスを、`!2001:db8:ca2e::fa4c` は 2001:db8:ca2e::fa4c 以外の任意の IP アドレスを指定します。

一連の IP アドレスを拒否するには、大かっこで囲んだ IP アドレスのリストの前に「!」記号を付けます。たとえば、`![192.168.1.1,192.168.1.5]` は 192.168.1.1 と 192.168.1.5 を除くすべての IP アドレスを定義します。

（注） IP アドレスのリストを否定するには、大カッコを使用する必要があります。

否定文字と一緒に IP アドレス リストを使用する場合は注意が必要です。たとえば、192.168.1.1 と 192.168.1.5 を除くすべてのアドレスと一致させるために `[!192.168.1.1,!192.168.1.5]` を使用した場合、システムはこの構文を「192.168.1.1 以外のすべて、**または** 192.168.1.5 以外のすべて」と解釈します。

192.168.1.5 は 192.168.1.1 ではなく、192.168.1.1 は 192.168.1.5 ではないため、この両方の IP アドレスが `[!192.168.1.1,!192.168.1.5]` という IP アドレス値に一致します。つまり、実質的に「`any`」を使用するのと同じです。

代わりに `![192.168.1.1,192.168.1.5]` を使用してください。システムはこの構文を「192.168.1.1 **でなく、しかも** 192.168.1.5 **でない**」と解釈し、大カッコ内に列挙されたものを除くすべての IP アドレスに一致します。

論理的に言って、`any` を除外（negation）と同時に使用できないことに注意してください。any を除外すると「アドレスなし」を意味することになります。

関連トピック

変数セット

侵入ルール ヘッダーの送信元および宛先ポート

侵入ルール エディタの [送信元ポート（Source Port）] フィールドと [宛先ポート（Destination Port）] フィールドで、送信元および宛先ポートを指定します。

侵入ルールのポート構文

ルールヘッダー内で使われるポート番号を定義するために、システムは特殊なタイプの構文を使用します。

（注） プロトコルが `ip` に設定されている場合、システムは侵入ルール ヘッダー内のポート定義を無視します。

次の例に示すように、カンマでポートを区切ることによって、ポートのリストを指定できます。

```
80, 8080, 8138, 8600-9000, !8650-8675
```

オプションで、次の例に示すように、ポートリストを大カッコで囲むこともできます（以前のソフトウェア バージョンではこれが必須でしたが、現在は必須ではありません）。

```
[80, 8080, 8138, 8600-9000, !8650-8675]
```

なお、次の例に示すように、ポートリストの否定を大カッコで囲む**必要がある**ことに注意してください。

```
![20, 22, 23]
```

次の表に、使用可能な構文を要約します。

表 *4*: 送信元/宛先ポート構文

指定する項目	用途	例
任意のポート	任意	任意
1 つの特定のポート	ポート番号	80

指定する項目	用途	例
ポートの範囲	範囲内の最初のポート番号と最後のポート番号をダッシュでつなぐ	`80-443`
1 つの特定のポートに等しい、またはより小さいすべてのポート	ポート番号の前にダッシュを付ける	`-21`
1 つの特定のポートに等しい、またはより大きいすべてのポート	ポート番号の後ろにダッシュを付ける	`80-`
1 つの特定のポートまたはポート範囲を除く、すべてのポート	否定する場合には、ポート、ポート リスト、ポート範囲の前に文字 ! を付けます。 否定が「ポートなし」を示す場合を除いて、すべてのポート宛先に論理上、否定を使用できる点にご注意ください。	`!20`
ポート変数で定義されるすべてのポート	`$` の後ろに英大文字の変数名	`$HTTP_PORTS`
ポート変数で定義されるポートを除く、すべてのポート	`!$` の後ろに英大文字の変数名	`!$HTTP_PORTS`

侵入イベント詳細

標準のテキスト ルールを作成するときには、ルールでエクスプロイト試行を検出する対象となる脆弱性についてのコンテキスト情報を含めることができます。また、脆弱性データベースへの外部参照を含めたり、組織内でイベントに設定するプライオリティを定義したりすることもできます。アナリストがイベントを認識すると、そのプライオリティ、エクスプロイト、および既知の対策についての情報をすぐに入手できます。

メッセージ

ルールのトリガー時にメッセージとして表示される、意味のあるテキストを指定できます。メッセージを読むと、ルールで攻撃試行を検出する対象となった脆弱性の特性をすぐに理解できます。中カッコ（{}）を除く、印字可能な任意の標準 ASCII 文字を使用できます。システムは、メッセージ全体を囲んでいる引用符を取り除きます。

ヒント ルール メッセージの指定は必須です。また、空白文字のみ、1 つ以上の引用符のみ、1 つ以上のアポストロフィのみ、あるいは空白文字/引用符/アポストロフィだけの組み合わせでメッセージを構成することはできません。

侵入ルール エディタでイベント メッセージを定義するには、[メッセージ（Message）] フィールドにイベント メッセージを入力します。

分類

ルールごとに、イベントのパケット表示に含める攻撃分類を指定できます。次の表に、それぞれの分類の名前と番号を示します。

表 *5:* ルールの分類

番号	分類名	説明
1	not-suspicious	不審ではないトラフィック
2	unknown	不明なトラフィック
3	bad-unknown	有害な可能性のあるトラフィック
4	attempted-recon	情報漏えいが試行された
5	successful-recon-limited	情報漏えいが発生
6	successful-recon-largescale	大規模な情報漏えい
7	attempted-dos	サービス拒否が試行された
8	successful-dos	サービス拒否が発生
9	attempted-user	ユーザ特権の獲得が試行された
10	unsuccessful-user	ユーザ特権の獲得が失敗した
11	successful-user	ユーザ特権の獲得に成功
12	attempted-admin	管理者特権の獲得が試行された
13	successful-admin	管理者特権の獲得に成功
18	rpc-portmap-decode	RPC クエリのデコード
15	shellcode-detect	実行可能コードが検出された
16	string-detect	疑わしい文字列が検出された
17	suspicious-filename-detect	疑わしいファイル名が検出された
18	suspicious-login	疑わしいユーザ名を使用したログイン試行が検出された
19	system-call-detect	システム コールが検出された
20	tcp-connection	TCP 接続が検出された
21	trojan-activity	ネットワーク トロイの木馬が検出された

番号	分類名	説明
22	unusual-client-port-connection	通常とは異なるポートをクライアントが使用していた
23	network-scan	ネットワーク スキャンの検出
24	denial-of-service	サービス拒否攻撃の検出
25	non-standard-protocol	非標準プロトコルまたはイベントの検出
26	protocol-command-decode	一般的なプロトコル コマンド デコード
27	web-application-activity	脆弱な可能性のある Web アプリケーションへのアクセス
36	web-application-attack	Web アプリケーション攻撃
29	misc-activity	その他のアクティビティ
30	misc-attack	その他の攻撃
31	icmp-event	一般的な ICMP イベント
32	inappropriate-content	不適切な内容が検出された
33	policy-violation	企業プライバシー侵害の可能性
34	default-login-attempt	デフォルトのユーザ名とパスワードによるログイン試行
35	sdf	機密データ
36	malware-cnc	既知のマルウェア コマンドと制御トラフィック
37	client-side-exploit	既知のクライアント側 exploit 試行
38	file-format	既知の悪意のあるファイルまたはファイルベースのエクスプロイト

カスタム分類

定義したルールによって生成されるイベントのパケット表示記述の内容をもっとカスタマイズする必要がある場合には、カスタム分類を作成できます。

引数	説明
分類名	分類の名前。40 文字を超える文字を使用すると、ページが読みにくくなります。<>()\'"&$; 文字および空白文字はサポートされていません。

引数	説明
分類の説明	分類の説明。英数字とスペースを使用できます。 <>()\'"&$; 文字はサポートされていません。
プライオリティ	高（High）、中（medium）、または低（low）。

カスタム プライオリティ

デフォルトでは、ルールのイベント分類からルールのプライオリティが派生します。ただし、`priority` キーワードをルールに追加し、高、中、または低のプライオリティを選択することで、ルールの分類優先度を上書きすることができます。たとえば、Webアプリケーション攻撃を検出するルールに高プライオリティを割り当てるには、`priority`キーワードをルールに追加して、プライオリティとして [高（high）] を選択します。

カスタム参照

`reference` キーワードを使用すると、イベントに関する外部 Web サイトや追加情報への参照を追加できます。参照を追加すると、アナリストは参照情報をすぐに利用できるため、パケットがルールをトリガーとして使用した理由を特定するのに役立ちます。次の表に、既知のエクスプロイトや攻撃についてのデータを提供する外部システムをいくつか示します。

表 *6 :* 外部攻撃識別システム

システム ID	説明	ID の例
`bugtraq`	[Bugtraq] ページ	`8550`
`cve`	共通脆弱性（CVE）ID	`2020-9607`
`mcafee`	[McAfee] ページ	`98574`
`url`	Web サイト参照	`www.example.com?exploit=14`
`msb`	Microsoftセキュリティ情報	`MS11-082`
`nessus`	[Nessus] ページ	`10039`
`secure-url`	セキュア Web サイト参照（https://...）	`intranet/exploits/exploit=14` 任意のセキュア Web サイトで `secure-url` を使用できることに注意してください。

次のように、参照値を入力して参照を指定します。

```
id_system,id
```

ここで、id_system はプレフィックスとして使用されるシステム、id は CVE ID 番号、Arachnids ID、または URL（http://なし）です。

たとえば、CVE-2020-9607 で文書化されている Adobe Acrobat および Reader の問題を指定するには、次の値を入力します。

```
cve,2020-9607
```

参照をルールに追加するときには、次の点に注意してください。

- カンマの後ろにスペースを入力しないでください。
- システム ID に大文字を使用しないでください。

関連トピック

カスタム分類の追加 （14 ページ）
イベント優先順位の定義 （15 ページ）
イベント参照の定義 （15 ページ）

カスタム分類の追加

手順

ステップ 1 ルールの作成または編集時に、[分類（Classification）] ドロップダウンリスト（**[オブジェクト（Objects）] > [侵入ルール（Intrusion Rules）] > [ルールの作成（Create Rules）] > [分類の編集（Edit Classifications）]**）から [分類の編集（Edit Classifications）] を選択します。

代わりに [分類の表示（View Classifications）] が表示される場合、設定は先祖ドメインに属しており、設定を変更する権限がありません。

ステップ 2 侵入イベント詳細 （10 ページ） の説明に従い、[分類名（Classification Name）] と [分類の説明（Classification Description）] を入力します。

ステップ 3 [優先度（Priority）] ドロップダウン リストから分類の優先度を選択します。

ステップ 4 [追加（Add）] をクリックします。

ステップ 5 [完了（Done）] をクリックします。

次のタスク

- ルールの作成または編集を続けます。詳細については、新規ルールの作成 （17 ページ） または 既存のルールの変更 （18 ページ） を参照してください。

関連トピック

カスタム ルールの作成 （16 ページ）

イベント優先順位の定義

手順

ステップ 1 ルールの作成または編集時に、[検出オプション（Detection Options）] ドロップダウン リストから [優先順位（priority）] を選択します。

ステップ 2 [Add Option] をクリックします。

ステップ 3 [優先順位（priority）] ドロップダウン リストから値を選択します。

ステップ 4 [保存（Save）] をクリックします。

次のタスク

- ルールの作成または編集を続けます。詳細については、新規ルールの作成 （17 ページ）または 既存のルールの変更 （18 ページ） を参照してください。

関連トピック

カスタム ルールの作成 （16 ページ）

イベント参照の定義

手順

ステップ 1 ルールの作成または編集時に、[検出オプション（Detection Options）] ドロップダウン リストから [参照（reference）] を選択します。

ステップ 2 [Add Option] をクリックします。

ステップ 3 侵入イベント詳細 （10 ページ）の説明に従って、[参照（reference）] フィールドに値を入力します。

ステップ 4 [保存（Save）] をクリックします。

次のタスク

- ルールの作成または編集を続けます。詳細については、新規ルールの作成 （17 ページ）または 既存のルールの変更 （18 ページ） を参照してください。

関連トピック

カスタム ルールの作成 （16 ページ）

カスタム ルールの作成

カスタム侵入ルールは以下の方法で作成できます。

- 独自の標準テキスト ルールを作成する
- 既存の標準テキスト ルールを新規ルールとして保存する
- システムが提供する共有オブジェクト ルールを新規ルールとして保存する
- ローカル ルール ファイルをインポートする

作成方法に関わらず、システムはカスタム ルールをローカル ルールに分類して保存します。

カスタム侵入ルールを作成すると、システムは一意のルール番号（番号の形式は`GID:SID:Rev`）を割り当てます。この番号には次の要素が含まれます。

GID

ジェネレータ ID。すべての標準テキスト ルールでは、この値は 1（グローバル ドメインまたはレガシー GID）または 1000 ～ 2000（子孫ドメイン）です。共有オブジェクト ルールを新規ルールとして保存する場合、値は 1 です。

SID

Snort ID。ルールがシステム ルールのローカル ルールであるかどうかを示します。新しいルールを作成すると、システムは次に使用可能なローカル ルール SID 番号を割り当てます。

ローカル ルールの SID 番号は 1000000 から始まり、新しいローカル ルールにつき番号が 1 ずつ増えます。

Rev

改訂番号。新しいルールのリビジョン番号は 1 です。カスタム ルールを変更するたびに、リビジョン番号が 1 ずつ増えます。

カスタム標準テキスト ルールでは、ルール ヘッダー設定、ルール キーワード、およびルール引数を設定できます。特定のプロトコルを使用する、特定の IP アドレスまたはポートを行き来するトラフィックだけをルールで照合するよう、ルール ヘッダーを設定できます。

システムが提供する標準テキスト ルールまたは共有オブジェクト ルールのカスタム ルールで変更できるルール ヘッダー情報は、送信元と宛先ポートと IP アドレスなどの情報に限られます。ルール キーワードやルール引数は変更できません。

共有オブジェクト ルールのヘッダー情報を変更して変更内容を保存すると、ルールの新しいインスタンスが作成され、ジェネレータ ID（GID） 1（グローバル ドメイン）または 1000 ～ 2000（子孫ドメイン）、およびカスタム ルールとして次に使用可能な SID が割り当てられます。システムは、共有オブジェクト ルールの新しいインスタンスを予約済み `soid` キーワードにリンクします。これにより、新しく作成したルールが Talos インテリジェンス グループ 作成のルールにマップされます。ユーザーが作成した共有オブジェクト ルールのインスタンスは削除できますが、Talos が作成した共有オブジェクト ルールは削除できません。

新規ルールの作成

手順

ステップ 1 **[オブジェクト（Objects）] > [侵入ルール（Intrusion Rules）]**を選択します。

ステップ 2 [Create Rule] をクリックします。

ステップ 3 [メッセージ（Message）] フィールドに値を入力します。

ステップ 4 次の各ドロップダウン リストから値を選択します。

- **[分類（Classification）]**
- **操作**
- **[Protocol]**
- **方向（Direction）**

ステップ 5 次のフィールドに値を入力します。

- **[送信元 IP（Source IPs）]**
- **[宛先 IP（Destination IPs）]**
- **送信元ポート（Source Port）**
- **宛先ポート（Destination Port）**

これらのフィールドに値を指定しない場合、システムは値 [すべて（any）] を使用します。

ステップ 6 [検出オプション（Detection Options）] ドロップダウン リストから値を選択します。

ステップ 7 [Add Option] をクリックします。

ステップ 8 追加したキーワードの引数を入力します。

ステップ 9 必要に応じて、手順 6 ～ 8 を繰り返します。

ステップ 10 複数のキーワードを追加した場合、以下を実行できます。

- キーワードの並べ替え：移動するキーワードの横にある上矢印または下矢印をクリックします。
- キーワードの削除：そのキーワードの横にある [X] をクリックします。

ステップ 11 [新規として保存（Save As New）] をクリックします。

次のタスク

- 該当する侵入ポリシー内の新規または変更されたルールを有効にします（侵入ポリシー内の侵入ルールの表示を参照）。
- 設定変更を展開します設定変更の展開を参照してください。

既存のルールの変更

システム提供のルールと先祖ドメインに属しているルールは、新しいカスタム ルールとしてローカル ルール カテゴリに保存してから変更できます。

手順

ステップ 1 次のいずれかの方法を使用して侵入ルールにアクセスします。

- **[ポリシー（Policies）] > [アクセス制御（Access Control）] > [侵入（Intrusion）]**を選択します。

  編集するポリシーの横にある [Snort2バージョン（Snort 2 Version）] をクリックし、[ルール（Rules）] をクリックします。

- **[オブジェクト（Objects）] > [侵入ルール（Intrusion Rules）]**を選択します。

ステップ 2 変更するルールを見つけます。次の選択肢があります。

- フォルダからルールに移動します。
- ルールを検索します。ルールの検索 （22 ページ）を参照してください。
- ルールが属しているグループにフィルタを適用します。フィルタリング ルール （26 ページ）を参照してください。

ステップ 3 ルールの横にある **[編集（Edit）]**（  ）をクリックします。検索結果の場合はルールメッセージをクリックします。

代わりに [表示（View）]（  ） が表示される場合、設定は先祖ドメインに属しており、設定を変更する権限がありません。

ステップ 4 ルール タイプに応じて、ルールを変更します。

（注） 共有オブジェクト ルールのプロトコルは変更しないでください。これを変更すると、ルールの効果がなくなる可能性があります。

ステップ 5 次の選択肢があります。

- カスタム ルールを編集していて、そのルールの現在のバージョンを上書きする場合は、[保存（Save）] をクリックします。
- システム提供のルールまたは先祖ドメインに属しているルールを編集している場合や、カスタム ルールを編集しているときに変更を新しいルールとして保存する場合は、[新規に保存（Save As New）] をクリックします。

次のタスク

- システム提供のルールの代わりにローカルで変更したルールを使用するには、侵入ルールの状態の手順に従ってシステム提供のルールを非アクティブ化してから、ローカル ルールをアクティブ化します。
- 設定変更を展開します設定変更の展開を参照してください。

関連トピック

ルールの検索 （22 ページ）
侵入ルール エディタ ページでのルールのフィルタリング （24 ページ）

ルール ドキュメンテーションの表示

[ルールの編集（Rule Edit）] ページから、Talos インテリジェンスグループ によって提供されるルール ドキュメンテーションを表示できます。 表示中に、**[ルールドキュメンテーション（Rule Documentation）]** およびその他の外部参照をクリックして、Talos によって提供される追加情報を表示できます。[コンテキスト エクスプローラ（Context Explorer）] をクリックして、ルールによって生成されたイベントのコンテキスト情報を表示することもできます。

手順

ステップ 1 次のいずれかの方法を使用して侵入ルールにアクセスします。

- **[ポリシー（Policies）]** > **[アクセス制御（Access Control）]** > **[侵入（Intrusion）]**を選択します。

  編集するポリシーの横にある [Snort2バージョン（Snort 2 Version）] をクリックし、[ルール（Rules）] をクリックします。

- **[オブジェクト（Objects）]** > **[侵入ルール（Intrusion Rules）]**を選択します。

ステップ 2 表示するルールを探します。次の選択肢があります。

- フォルダからルールに移動します。
- ルールを検索します。ルールの検索 （22 ページ）を参照してください。
- ルールが属しているグループにフィルタを適用します。フィルタリング ルール （26 ページ）を参照してください。

ステップ 3 ルールの横にある **[編集（Edit）]**（✎）をクリックします。検索結果の場合はルールメッセージをクリックします。

代わりに [表示（View）]（👁） が表示される場合、設定は先祖ドメインに属しており、設定を変更する権限がありません。

ステップ 4 [ドキュメントの表示（View Documentation）] をクリックします。

ステップ 5 状況に応じて、次のいずれかのリンクをクリックします。

- [ルール ドキュメンテーション（Rule Documentation）]：ルール固有の詳細を表示します。
- [その他の外部参照（Other external references）]：利用可能な外部参照に関する情報については、キーワード フィルタリング （24 ページ）および侵入イベント詳細 （10 ページ）の「カスタム リファレンス」を参照してください。
- [コンテキストエクスプローラ（Context Explorer）]：コンテキスト エクスプローラでのルールのコンテキストデータの表示に関する情報については、Cisco Secure Firewall Management Center アドミニストレーション ガイド の侵入情報セクションを参照してください。

ヒント 外部リンクを選択するとドキュメンテーションのポップアップ ウィンドウが閉じます。ルールを変更せずにルール編集ページを終了するには、任意のメニューパスを選択します。

侵入ルールへのコメントの追加

任意の侵入ルールにコメントを追加できます。コメントにより、環境や条件の説明と、ルールやルールが検出する悪意あるプログラム、スクリプト（エクスプロイト）やポリシー違反の詳細を示すことができます。

手順

ステップ 1 次のいずれかの方法を使用して侵入ルールにアクセスします。

- **[ポリシー（Policies）] > [アクセス制御（Access Control）] > [侵入（Intrusion）]**を選択します。

  編集するポリシーの横にある [Snort2バージョン（Snort 2 Version）] をクリックし、[ルール（Rules）] をクリックします。

- **[オブジェクト（Objects）] > [侵入ルール（Intrusion Rules）]**を選択します。

ステップ 2 注釈を付けるルールを探します。次の選択肢があります。

- フォルダからルールに移動します。
- ルールを検索します。ルールの検索 （22 ページ）を参照してください。
- ルールが属するグループをフィルタします。フィルタリング ルール （26 ページ） を参照してください。

ステップ 3 ルールの横にある **[編集（Edit）]**（  ）をクリックします。検索結果の場合はルールメッセージをクリックします。

代わりに [表示（View）]（  ）がルールの横に表示される場合、ルールは先祖ポリシーに属しており、ルールを変更する権限がありません。

ステップ 4 [ルールのコメント（Rule Comment）] をクリックします。

ステップ 5 テキスト ボックスにコメントを入力します。

ステップ 6 [コメントを追加（Add a Comment）] をクリックします。

ヒント また、侵入イベントのパケット ビューで、ルール コメントを追加して表示することもできます。

次のタスク

- ルールの作成または編集を続けます。詳細については、新規ルールの作成 （17 ページ）または 既存のルールの変更 （18 ページ） を参照してください。

関連トピック

ルールの検索 （22 ページ）

カスタム ルールの削除

侵入ポリシーで現在有効になっていないカスタム ルールを削除することができます。システムにより提供されている標準テキスト ルールおよび共有オブジェクト ルールは削除できません。

削除されたルールは削除済みカテゴリに保存されます。削除済みのルールを、新しいルールの基準として使用することができます。侵入ポリシーの [Rules] ページには削除済みカテゴリが表示されないため、削除したカスタム ルールを有効にすることはできません。

ヒント カスタム ルールには、変更されたヘッダー情報で保存する共有オブジェクト ルールが含まれます。また、これらはローカル ルール カテゴリに保存され、1（グローバル ドメインまたはレガシー GID）または 1000 ～ 2000（子孫ドメイン）の GID を使用してリストされます。変更した共有オブジェクト ルールは削除できますが、元の共有オブジェクト ルールは削除できません。

手順

ステップ 1 次のいずれかの方法を使用して侵入ルールにアクセスします。

- **[ポリシー（Policies）] > [アクセス制御（Access Control）] > [侵入（Intrusion）]**を選択します。

  編集するポリシーの横にある [Snort2バージョン（Snort 2 Version）] をクリックし、[ルール（Rules）] をクリックします。

- **[オブジェクト（Objects）] > [侵入ルール（Intrusion Rules）]**を選択します。

ステップ 2 次の 2 つの選択肢があります。

- すべてのローカル ルールを削除します：[ローカル ルールの削除（Delete Local Rules）] をクリックし、[OK] をクリックします。
- 1 つのルールを削除します：[ルールのグループ化基準（Group Rules By）] ドロップダウンから [ローカルルール（Local Rules）] を選択し、削除するルールの隣にある [削除（Delete）]（🗑） をクリックし、[OK] をクリックして削除を確認します。

関連トピック

侵入ルールの状態

ルールの検索

システムには、数千もの標準テキストルールが用意されています。また、Talos インテリジェンスグループは新しい脆弱性およびエクスプロイトが見つかったときのルールを追加を継続します。特定のルールを簡単に検索して、そのルールをアクティブ化、非アクティブ化、または編集することができます。

手順

ステップ 1 次のいずれかの方法を使用して侵入ルールにアクセスします。

- **[ポリシー（Policies）] > [アクセス制御（Access Control）] > [侵入（Intrusion）]**を選択します。

  編集するポリシーの横にある [Snort2バージョン（Snort 2 Version）] をクリックし、[ルール（Rules）] をクリックします。

- **[オブジェクト（Objects）] > [侵入ルール（Intrusion Rules）]**を選択します。

ステップ 2 ツールバーで [検索（Search）] をクリックします。

ステップ 3 検索条件を追加します。

ステップ 4 [検索（Search）] をクリックします。

次のタスク

- 見つかったルール（システム ルールの場合はルールのコピー）を表示または編集する場合は、ハイパーリンクが付いたルール メッセージをクリックします。詳細については、新規ルールの作成 （17 ページ） または 既存のルールの変更 （18 ページ） を参照してください。

侵入ルールの検索条件

次の表には、利用可能な検索オプションについて説明しています。

表 *7:* ルール検索規則

オプション	説明
署名 ID（Signature ID）	Snort ID（SID）に基づいて 1 つのルールを検索するには、SID 番号を入力します。複数のルールを検索するには、SID 番号リストをコンマで区切って入力します。このフィールドは、80 文字以内です。
ジェネレータ ID（Generator ID）	標準テキスト ルールを検索するには、[1] を選択します。共有オブジェクトのルールを検索するには、[3] を選択します。
メッセージ（Message）	特定のメッセージを含むルールを検索するには、ルール メッセージの 1 つの単語を [メッセージ（Message）] フィールドに入力します。たとえば、DNS exploit を検索するには「`DNS`」と入力し、バッファ オーバーフローエクスプロイトを検索するには「`overflow`」と入力します。
プロトコル（Protocol）	特定のプロトコルのトラフィックを評価するルールを検索するには、そのプロトコルを選択します。プロトコルを選択しない場合、検索結果にはすべてのプロトコルのルールが含まれます。
送信元ポート（Source Port）	指定ポートから発信されるパケットを調べるルールを検索するには、送信元ポート番号またはポート関連変数を入力します。
接続先ポート（Destination Port）	特定ポートを宛先にしたパケットを調べるルールを検索するには、宛先ポート番号かポート関連変数を入力します。
ソース IP（Source IP）	特定の IP アドレスから発信されるパケットを調べるルールを検索するには、送信元 IP アドレスまたは IP アドレス関連変数を入力します。
宛先 IP（Destination IP）	特定の IP アドレスに送信するパケットを調べるルールを検索するには、宛先 IP アドレスまたは IP アドレス関連変数を入力します。
キーワード（Keyword）	特定のキーワードを検索するには、キーワード検索オプションを使用できます。キーワードを選択して、検索するキーワード値を入力します。特定値以外の任意の値に一致させるには、キーワードの前に疑問符（`!`）を入力します。
カテゴリ（Category）	特定のカテゴリ内のルールを検索するには、[カテゴリ（Category）] リストからカテゴリを選択します。
分類（Classification）	特定の分類が設定されたルールを検索するには、[分類（Classification）] リストから分類名を選択します。
ルール状態（Rule State）	特定のポリシー内のルールや特定のルール状態を検索するには、最初のルール状態リストからポリシーを選択し、第 2 のリストから状態を選択して、イベントの作成、イベントのドロップ、作成、無効に設定されたルールを検索します。

侵入ルール エディタ ページでのルールのフィルタリング

侵入ルール エディタ ページ上でルールをフィルタリングして、ルールのサブセットを表示することができます。たとえば、あるルールまたはその状態を変更したいが、数千ものルールの中からそれを見つけるのが困難な場合に、この機能が役立つことがあります。

フィルタを入力すると、1 つ以上の一致するルールを含むフォルダがページに表示され、一致するルールがない場合はメッセージが表示されます。

フィルタリング ガイドライン

フィルタには、特殊なキーワードとその引数、文字列、引用符で囲んだリテラル文字列、さらに複数のフィルタ条件を区切るスペースを含めることができます。ただし、正規表現、ワイルドカード文字、および否定文字（!）、「大なり」記号（>）、「小なり」記号（<）などの特殊な演算子をフィルタに含めることはできません。

すべてのキーワード、キーワード引数、および文字列では大文字と小文字が区別されません。`gid` キーワードと `sid` キーワードを除くすべての引数と文字列が部分文字列として扱われます。`gid` と `sid` の引数は、完全一致のみを返します。

フィルタ処理前の元のページで 1 つのフォルダを展開すると、その後のフィルタ処理でそのフォルダ内の一致が返されるときにフォルダが展開したままになります。探しているルールが多数のルールを含むフォルダ内に存在する場合には、これが役立つことがあります。

1 つのフィルタを後続の別のフィルタで制約することはできません。入力されたフィルタは、ルール データベース全体を検索して、一致するすべてのルールを返します。前回のフィルタ結果がページに表示されている状態でフィルタを入力すると、そのページの内容が消去され、代わりに新しいフィルタの結果が返されます。

フィルタ処理されている場合もされていない場合も、リスト内のルールで同じ機能を使用できます。たとえば、侵入ルール エディタ ページでは、リストがフィルタ処理されているかどうかに関わらず、リスト内のルールを編集できます。また、ページのコンテキスト メニューの任意のオプションを使用することもできます。

ヒント すべてのサブグループ内のルールの合計数が多い場合は、フィルタリングに長い時間がかかることがあります。これは、個別のルールの数がかなり少なくても、1 つのルールが複数のカテゴリに出現することがあるためです。

キーワード フィルタリング

各ルール フィルタに、次の形式で 1 つ以上のキーワードを含めることができます。

```
keyword:argument
```

ここで、keyword は次の表のいずれかのキーワード、argument はキーワードに関連する特定のフィールドで検索される単一の、大文字/小文字を区別しない英数字文字列です。

gid と sid を除くすべてのキーワードの引数が部分文字列として扱われます。たとえば、引数 123 によって "12345"、"41235"、"45123" などが返されます。gid と sid の引数は完全一致のみを返します。たとえば、sid:3080 によって SID 3080 のみが返されます。

ヒント　部分的な SID を検索するには、1 つ以上の文字列を使ってフィルタ処理できます。

次の表に、ルールのフィルタ処理に使用できる特定のフィルタリング キーワードと引数を示します。

表 *8 :* ルール フィルタ キーワード

キーワード	説明	例
arachnids	ルール参照内の Arachnids ID 全体またはその一部分に基づいて 1 つ以上のルールを返します。	arachnids:181
bugtraq	ルール参照内の Bugtraq ID 全体またはその一部分に基づいて 1 つ以上のルールを返します。	bugtraq:2120
cve	ルール参照内の CVE 番号全体またはその一部分に基づいて 1 つ以上のルールを返します。	cve:2003-0109
gid	引数 1 は標準のテキスト ルールを返します。引数 3 は共有オブジェクト ルールを返します。	gid:3
mcafee	ルール参照内の McAfee ID 全体またはその一部分に基づいて 1 つ以上のルールを返します。	mcafee:10566
msg	ルールの [メッセージ（Message）] フィールド（イベント メッセージとも呼ばれる）の全体またはその一部分に基づいて 1 つ以上のルールを返します。	msg:chat
nessus	ルール参照内の Nessus ID 全体またはその一部分に基づいて 1 つ以上のルールを返します。	nessus:10737
ref	ルール参照内またはルールの [メッセージ（Message）] フィールド内の単一の英数字文字列の全体または一部分に基づいて、1 つ以上のルールを返します。	ref:MS03-039
sid	正確な Snort ID を持つルールを返します。	sid:235
url	ルール参照内の URL 全体またはその一部分に基づいて 1 つ以上のルールを返します。	url:faqs.org

関連トピック

イベント参照の定義 （15 ページ）
侵入イベント詳細 （10 ページ）

文字列フィルタリング

各ルールフィルタに1つ以上の英数字文字列を含めることができます。文字列により、ルールの [メッセージ（Message）] フィールド、Snort ID ID（SID）、およびジェネレータ ID が検索されます。たとえば、文字列 `123` を指定すると、ルール メッセージ内の文字列「`Lotus123`」や「`123mania`」などが返され、さらに、SID 6123、SID 12375 なども返されます。

すべての文字列では大文字と小文字が区別されず、部分的な文字列として扱われます。たとえば、文字列 `ADMIN`、`admin`、または `Admin` はすべて、「`admin`」、「`CFADMIN`」、「`Administrator`」などを返します。

文字列を引用符で囲むと、完全一致を返すことができます。たとえば、引用符付きのリテラル文字列 `"overflow attempt"` は完全一致のみを返しますが、引用符なしの2つの文字列 `overflow` と `attempt` で構成されるフィルタは `"overflow attempt"`、`"overflow multipacket attempt"`、`"overflow with evasion attempt"` などを返します。

関連トピック

侵入イベント詳細 （10 ページ）

キーワードと文字列の組み合わせによるフィルタリング

複数のキーワード、文字列、またはその両方をスペースで区切って任意に組み合わせて入力することで、フィルタリングの結果を絞り込むことができます。結果には、すべてのフィルタ条件に一致するルールが含まれます。

複数のフィルタ条件を任意の順序で入力できます。たとえば、次のフィルタはそれぞれ同じルールを返します。

- `url:at login attempt cve:200`
- `login attempt cve:200 url:at`
- `login cve:200 attempt url:at`

フィルタリング ルール

[侵入ルール（Intrusion Rules）] ページで、ルールをサブセットにフィルタ処理すると、より簡単に特定のルールを見つけることができます。その後で、いずれかのページ機能を使用できます。これには、コンテキスト メニューで使用可能な機能の選択も含まれます。

編集する特定のルールを見つけるのに、規則のフィルタリングはとても役立ちます。

手順

ステップ 1 次のいずれかの方法を使用して侵入ルールにアクセスします。

- **[ポリシー（Policies）]** > **[アクセス制御（Access Control）]** > **[侵入（Intrusion）]**を選択します。

  編集するポリシーの横にある [Snort2バージョン（Snort 2 Version）] をクリックし、[ルール（Rules）] をクリックします。

- **[オブジェクト（Objects）]** > **[侵入ルール（Intrusion Rules）]**を選択します。

ステップ 2 フィルタリングする前に、次の選択を行います。

- 該当のルール グループを展開します。複数のルール グループにも、展開できるサブグループがあります。

  また、ルールがどのグループに含まれているか予想できる場合は、フィルタ処理前の元のページでそのグループを展開しておくと便利な場合があります。その後のフィルタ処理でそのフォルダ内の一致した結果が返されるとき、およびフィルタ [クリア（Clear）]（✕）をクリックしてフィルタ処理前のページに戻ったときに、グループが展開されたままになります。

- [グループ ルール（Group Rules By）] ドロップダウンリストから別のグループ メソッドを選択します。

ステップ 3 [グループルール（Group Rules By）] リストで [フィルタ（Filter）]（🔍） の横にあるテキストボックスにフィルタ制約を入力します。

ステップ 4 Enter を押します。

（注） フィルタ [クリア（Clear）]（✕） をクリックして、現在のフィルタ処理されたリストをクリアします。

侵入ルールのキーワードと引数

ルール言語では、キーワードを組み合わせることによってルールの動作を指定できます。キーワードとそれに関連する値（引数と呼ばれる）は、ルールエンジンによって検査されるパケットおよびパケット関連値をシステムがどのように評価するかを決定します。システムでは現在、コンテンツマッチング、プロトコル固有のパターンマッチング、状態固有のマッチングなどのインスペクション機能を実行するためのキーワードがサポートされています。キーワードあたり最大100個の引数を定義し、互換性のある任意の数のキーワードを組み合わせて非常に具体的なルールを作成できます。これにより、誤検出や検出漏れの可能性が減少し、受け取った侵入情報に集中的に取り組むことができます。

また、パッシブ展開でadaptive profile updatesを使用すると、ルール メタデータとホスト情報に基づいて特定のパケットに対するアクティブ ルール処理を動的に調整できます。

ここに記載されているキーワードは、ルールエディタの検出オプションとして表示されます。

関連トピック

アダプティブ プロファイルについて

content キーワードと protected_content キーワード

content キーワードまたは protected_content キーワードを使用すると、パケット内から検出するコンテンツを指定できます。

ほとんどの場合、content または protected_content キーワードの後ろに修飾子を付けて、コンテンツを検索すべき場所、検索で大文字/小文字を区別するかどうか、およびその他のオプションを指定する必要があります。

ルールでイベントがトリガーとして使用されるためには、すべてのコンテンツ マッチングが真でなければならないことに注意してください。つまり、各コンテンツ マッチングは相互にAND関係にあります。

また、インライン展開では、有害なコンテンツを照合した後でそれを同じ長さの独自のテキスト文字列に置き換えるルールをセットアップできることにも注意してください。

content

content キーワードを使用すると、ルール エンジンはパケット ペイロードまたはストリームでその文字列を検索します。たとえば、いずれかの content キーワードの値として /bin/sh と入力した場合、ルール エンジンはパケット ペイロード内で文字列 /bin/sh を検索します。

ASCII 文字列、16 進コンテンツ（バイナリ バイト コード）、またはその両方の組み合わせを使用してコンテンツを照合できます。キーワード値の中で 16 進コンテンツをパイプ文字（|）で囲みます。たとえば、|90C8 C0FF FFFF|/bin/sh のように 16 進コンテンツと ASCII コンテンツを混在させることができます。

1 つのルール内で複数のコンテンツ マッチングを指定できます。これを行うには、content キーワードの追加のインスタンスを使用します。コンテンツ マッチングごとに、ルールをトリガーとして使用させるにはパケット ペイロードまたはストリームでコンテンツ一致が見つからなければならないことを指定できます。

注意 **Not** オプションが選択された 1 つの content キーワードだけを含むルールを作成した場合、侵入ポリシーの効果がなくなる可能性があります。

protected_content

protected_content キーワードを使用すると、ルール引数を設定する前に、検索コンテンツ文字列をエンコードすることができます。キーワードを設定する前に、ルール作成者がハッシュ関数（SHA-512、SHA-256、または MD5）を使用して文字列をエンコードします。

content キーワードの代わりに protected_content キーワードを使用した場合でも、ルール エンジンがパケット ペイロードまたはストリームの中で文字列を検索する方法に違いはなく、ほとんどのキーワード オプションが想定どおりに機能します。次の表は、protected_content キーワード オプションと content キーワード オプションの間の例外的な相違点を要約しています。

表 *9 : protected_content* オプションの例外

オプション	説明
ハッシュ タイプ（Hash Type）	protected_content ルール キーワードの新しいオプション。
大文字小文字の区別なし（Case Insensitive）	未サポート
次の範囲内（Within）	未サポート
奥行き（Depth）	未サポート
長さ（Length）	protected_content ルール キーワードの新しいオプション。
高速パターンマッチ機能を使用（Use Fast Pattern Matcher）	未サポート
高速パターンマッチ機能のみ（Fast Pattern Matcher Only）	未サポート
高速パターン マッチ機能オフセットおよび長さ（Fast Pattern Matcher Offset and Length）	未サポート

Cisco では、protected_content キーワードを含むルールに 1 つ以上の content キーワードを含めることを推奨しています。こうすると、ルール エンジンが常に高速パターン マッチ機能を使用することで処理速度が上がり、パフォーマンスが向上します。ルール内の protected_content キーワードの前に content キーワードを配置します。ルールに 1 つ以上の content キーワードが含まれている場合は、content キーワードの Use Fast Pattern Matcher 引数が有効になっているかどうかに関係なく、ルール エンジンが高速パターン マッチ機能を使用することに注意してください。

注意 **Not** オプションが選択された 1 つの protected_content キーワードだけを含むルールを作成した場合、侵入ポリシーの効果がなくなる可能性があります。

関連トピック

カスタム ルールの作成 （16 ページ）
基本コンテンツおよび protected_content キーワードの引数 （30 ページ）
replace キーワード （41 ページ）

基本コンテンツおよび protected_content キーワードの引数

content または protected_content キーワードを変更するパラメータを使用すると、コンテンツ検索の位置や大文字/小文字の区別を制約できます。content または protected_content キーワードを変更するオプションを設定して、検索対象となるコンテンツを指定します。

大文字小文字の区別なし（Case Insensitive）

（注） このオプションは protected_content キーワードの設定では**サポートされません**。

ASCII 文字列でコンテンツ一致を検索するときに大文字/小文字の区別を無視するようルールエンジンに指示できます。検索で大文字と小文字を区別しないようにするには、コンテンツ検索の指定時に [大文字小文字の区別なし（Case Insensitive）] をオンにします。

ハッシュ タイプ（Hash Type）

（注） このオプションは protected_content キーワードで**のみ**設定できます。

[ハッシュ タイプ（Hash Type）] ドロップダウンを使用して、検索文字列のエンコードに使用されたハッシュ関数を特定します。protected_content 検索文字列のハッシュ方式として、SHA-512、SHA-256、および MD5 がサポートされています。選択したハッシュ タイプとハッシュされたコンテンツの長さが一致しない場合、システムはルールを保存**しません**。

自動的に Cisco 設定のデフォルト値が選択されます。[デフォルト（Default）] が選択される場合、ルールに特定のハッシュ関数は含まれず、SHA-512 がハッシュ関数であると見なされます。

raw データ（Raw Data）

[raw データ（Raw Data）] オプションを使用すると、ルール エンジンは、正規化されたペイロードデータ（ネットワーク分析ポリシーによってデコードされたデータ）を分析する前にオリジナルのパケットペイロードを分析します。引数値は使用されません。正規化の前に、ペイロード内の Telnet ネゴシエーション オプションを検査するために Telnet トラフィックを分析する場合に、このキーワードを使用できます。

同じ content または protected_content キーワードで、**Raw Data** オプションを HTTP コンテンツ オプションと一緒に使用することはできません。

ヒント HTTP トラフィックで raw データを検査するかどうか、また、どの程度の量の raw データを検査するかを決定するため、HTTP 検査プリプロセッサの [クライアント フローの深さ（Client Flow Depth）] オプションと [サーバー フローの深さ（Server Flow Depth）] オプションを設定することができます。

注

指定したコンテンツと一致しないコンテンツを検索するには、[一致しない（Not）] オプションを選択します。[一致しない（Not）] オプションが選択された `content` または `protected_content` キーワードを含むルールを作成する場合には、そのルール内に、[一致しない（Not）] オプションが選択されていない別の `content` または `protected_content` キーワードを 1 つ以上含める必要があります。

注意 `content` または `protected_content` キーワードに対して **Not** オプションを選択した場合は、そのキーワードだけを含むルールを作成しないでください。侵入ポリシーの効果がなくなる可能性があります。

たとえば、SMTP ルール 1:2541: 9 に 3 つの `content` キーワードが含まれており、そのうちの 1 つで [一致しない（Not）] オプションが選択されているとします。[一致しない（Not）] オプションが選択されているキーワード以外のすべての `content` キーワードを削除すると、このルールに基づくカスタム ルールが無効になります。このようなルールを侵入ポリシーに追加すると、そのポリシーの効果がなくなる可能性があります。

ヒント 同じ `content` キーワードで、[Not] チェック ボックスと [Use Fast Pattern Matcher] チェック ボックスを同時に選択することはできません。

コンテンツ（content）および保護コンテンツ（protected_content）キーワード検索位置

検索位置オプションを使用すると、指定したコンテンツの検索をどこから開始するか、どこまで検索するかを指定できます。

許可された組み合わせ：content 検索位置の引数

次のように、2 つの `content` 位置ペアのいずれかを使用すると、指定したコンテンツの検索をどこから開始するか、どこまで検索するかを指定できます。

- パケット ペイロードの先頭を基準にして検索する場合は、**Offset** と **Depth** を一緒に使用します。
- 現在の検索位置を基準にして検索する場合は、**Distance** と **Within** を一緒に使用します。

ペアに含まれるオプションのどちらか 1 つだけを指定した場合は、そのペアのもう 1 つのオプションのデフォルトが想定されます。

Offset および **Depth** オプションと、**Distance** および **Within** オプションを混合することはできません。たとえば、**Offset** と **Within** をペアにすることはできません。1 つのルール内で任意の数の位置オプションを使用できます。

位置が指定されない場合は、**Offset** と **Depth** のデフォルトが想定されます。つまり、コンテンツ検索はパケット ペイロードの先頭から始まってパケットの末尾まで続きます。

また、既存の `byte_extract` 変数を使用して位置オプションの値を指定することもできます。

ヒント　1 つのルール内で任意の数の位置オプションを使用できます。

関連トピック

byte_extract キーワード （48 ページ）

許可された組み合わせ：protected_content 検索位置の引数

次のように、必須の **Length** `protected_content` 位置オプションを **Offset** または **Distance** 位置オプションと組み合わせて使用すると、指定されたコンテンツの検索をどこから開始するか、どこまで検索するかを指定できます。

- パケット ペイロードの先頭を基準にして、保護された文字列を検索するには、**Length** と **Offset** を一緒に使用します。
- 現在の検索位置を基準にして、保護された文字列を検索するには、**Length** と **Distance** を一緒に使用します。

ヒント　1 つのキーワード設定内で [オフセット（Offset）] オプションと [距離（Distance）] オプションを併用することはできませんが、1 つのルール内では任意の数の位置オプションを使用できます。

位置が指定されない場合は、デフォルトが想定されます。つまり、コンテンツ検索はパケット ペイロードの先頭から始まってパケットの末尾まで続きます。

また、既存の `byte_extract` 変数を使用して位置オプションの値を指定することもできます。

関連トピック

byte_extract キーワード （48 ページ）

content および protected_content の検索位置の引数

奥行

（注）　このオプションは、`content` キーワードを設定する場合に**のみ**サポートされます。

オフセット値の先頭からの（またはオフセットが設定されていない場合はパケットペイロード先頭からの）コンテンツ検索の最大の深さをバイト単位で指定します。

たとえば、ルールのコンテンツ値が `cgi-bin/phf`、`offset` 値が `3`、`depth` 値が `22` である場合、ルール ヘッダーで指定されたパラメータを満たすパケット内で、`cgi-bin/phf` 文字列との一致の検索がバイト位置 3 から始まり、22 バイト処理した後（バイト位置 25 で）停止します。

指定したコンテンツの長さ以上の、最大 65535 バイトまでの値を指定する必要があります。値 0 は指定できません。

デフォルトの深さは、「パケットの末尾まで検索」です。

距離（Distance）

以前に見つかったコンテンツ一致から数えて、指定されたバイト数の後に出現する後続のコンテンツ一致を見つけるようルール エンジンに指示します。

Distance（距離）カウンタはバイト 0 から始まるため、最後に見つかったコンテンツ一致から順方向に移動すべきバイト数よりも 1 つ少ない数値を指定してください。たとえば 4 を指定した場合、5 番目のバイトから検索が始まります。

-65535 ～ 65535 バイトを値として指定できます。負の `Distance` 値を指定した場合は、検索を開始するバイト位置がパケットの先頭から外れる可能性があります。実際にはパケットの第 1 バイトから検索が開始されますが、計算ではパケットの外側のバイトも考慮されます。たとえば、パケット内の現在の位置が第 5 バイトで、次のコンテンツ ルール オプションで `Distance` 値 -10 および `Within` 値 20 が指定された場合、検索はペイロードの先頭から開始され、[Within] オプションが 15 に調整されます。

デフォルトの距離は 0 で、これは最後のコンテンツ一致の後のパケット内の現在位置という意味です。

長さ（Length）

（注） このオプションは `protected_content` キーワードを設定する場合に**のみ**サポートされます。

Length `protected_content` キーワードオプションは、ハッシュされていない検索文字列の長さをバイト単位で示します。

たとえば、コンテンツ `Sample1` を使ってセキュア ハッシュを生成した場合には、**Length** 値として `7` を使用します。このフィールドに値を入力することは**必須**です。

オフセット（Offset）

パケット ペイロードの先頭を基準とする、コンテンツの検索を開始するパケット ペイロード内の位置をバイト単位で指定します。65535 ～ 65535 バイトを値として指定できます。

オフセット カウンタはバイト 0 から始まるため、パケット ペイロードの先頭から順方向に移動すべきバイト数よりも 1 つ少ない数値を指定してください。たとえば 7 を指定した場合は、8 番目のバイトから検索が始まります。

デフォルトのオフセットは 0 で、これはパケットの先頭を意味します。

次の範囲内（Within）

（注） このオプションは、`content` キーワードを設定する場合に**のみ**サポートされます。

[次の範囲内（Within）] オプションを使用すると、ルールをトリガーとして使用させるには、最後に見つかったコンテンツ一致の末尾以降、指定のバイト数以内に次のコンテンツ一致が発生する必要があることを指示できます。たとえば **Within** 値として `8` を指定した場合、次のコンテンツ一致がパケットペイロードの次の 8 バイト以内に発生する必要があります。発生しない場合は、ルールをトリガーとして使用する基準が満たされません。

指定したコンテンツの長さ以上の、最大 65535 バイトまでの値を指定できます。

[次の範囲内（Within）] のデフォルトは「パケットの末尾まで検索」です。

概要：HTTP content および protected_content キーワードの引数

HTTP `content` または `protected_content` キーワード オプションを使用すると、HTTP Inspect プリプロセッサによってデコードされた HTTP メッセージ内の一致コンテンツを検索する位置を指定できます。

次の 2 つのオプションは、HTTP 応答内のステータス フィールドを検索します。

- **HTTP ステータス コード（HTTP Status Code）**
- **HTTP ステータス メッセージ（HTTP Status Message）**

ルール エンジンは未加工の正規化されていないステータス フィールドを検索しますが、ここでは、他の Raw HTTP フィールドと正規化された HTTP フィールドを併用する際に考慮すべき制限についての説明を簡略化するために、これらのオプションが別個に列挙されていることに注意してください。

次の 5 つのオプションは、必要に応じて HTTP 要求、応答、またはその両方の中で正規化フィールドを検索します。

- **HTTP URI**
- **HTTP メソッド（HTTP Method）**
- **HTTP ヘッダー（HTTP Header）**
- **HTTP Cookie**
- **HTTP クライアント ボディ（HTTP Client Body）**

次の 3 つのオプションは、必要に応じて HTTP 要求、応答、またはその両方の中で未加工の（正規化されていない）非ステータス フィールドを検索します。

- **HTTP Raw URI**
- **HTTP Raw ヘッダー（HTTP Raw Header）**
- **HTTP Raw Cookie**

HTTP `content` オプションを選択する場合は、次のガイドラインに従ってください。

- HTTP `content` オプションは TCP トラフィックにのみ適用されます。
- パフォーマンスへの悪影響を避けるために、指定したコンテンツが出現する可能性のあるメッセージ部分だけを選択してください。

たとえば、ショッピング カート メッセージの場合のように大きな cookie がトラフィックに含まれている可能性がある場合は、HTTP cookie ではなく HTTP ヘッダーの中で指定のコンテンツを検索することができます。

- HTTP Inspect プリプロセッサの正規化機能を活用し、パフォーマンスを向上させるには、作成するすべての HTTP 関連ルールの中に少なくとも 1 つの `content` または `protected_content` キーワードを含め、それに対して **HTTP URI**、**HTTP Method**、**HTTP Header**、または **HTTP Client Body** オプションを選択します。

- HTTP `content` または `protected_content` キーワード オプションと組み合わせて `replace` キーワードを使用することはできません。

単一の正規化された HTTP オプションまたはステータス フィールドを指定できます。または、複数の正規化 HTTP オプションとステータス フィールドを任意に組み合わせて、コンテンツ領域をマッチング対象にすることもできます。ただし、HTTP フィールド オプションを使用する場合には次の制限事項に注意してください。

- 同じ `content` または `protected_content` キーワード内で、**Raw Data** オプションと HTTP オプションを一緒に使用することはできません。

- Raw HTTP フィールド オプション（[HTTP　Raw URI]、[HTTP Raw ヘッダー（HTTP Raw Header）]、または [HTTP Raw Cookie]）と、それぞれに対応する正規化されたオプション（[HTTP URI]、[HTTP ヘッダー（HTTP Header）]、または [HTTP Cookie]）を同じ `content` または `protected_content` キーワード内で一緒に使用することはできません。

- **Use Fast Pattern Matcher** を、次の 1 つ以上の HTTP フィールド オプションと組み合わせて選択することはできません。

  [HTTP Raw URI]、[HTTP raw ヘッダー（HTTP Raw Header）]、[HTTP Raw Cookie]、[HTTP Cookie]、[HTTP メソッド（HTTP Method）]、[HTTP ステータス メッセージ（HTTP Status Message）]、[HTTP ステータス コード（HTTP Status Code）]

  ただし、次のいずれかの正規化フィールドを検索するために高速パターンマッチ機能を使用する `content` または `protected_content` キーワードでは、上記のオプションを含めることができます。

  [HTTP URI]、[HTTP ヘッダー（HTTP Header）]、または [HTTP クライアント ボディ（HTTP Client Body）]

  たとえば、[HTTP Cookie]、[HTTP Header]、および [Use Fast Pattern Matcher] を選択した場合、ルール エンジンは HTTP cookie と HTTP ヘッダーの両方でコンテンツを検索しますが、高速パターン マッチ機能は HTTP cookie ではなく、HTTP ヘッダーにのみ適用されます。

- 制限付きオプションと制限なしオプションを併用した場合、高速パターンマッチ機能は、指定された制限なしフィールドのみを検索することで、侵入ルール エディタにルールを渡して（制限付きフィールドの評価を含む）完全な評価を行うべきかどうかを検査します。

関連トピック

content キーワードの高速パターン マッチ機能の引数 （39 ページ）

HTTP コンテンツと protected_content キーワードの引数

HTTP URI

正規化された要求URIフィールド内でコンテンツ一致を検索するには、このオプションを選択します。

このオプションと pcre キーワードの HTTP URI（U）オプションを一緒に使用して、同じコンテンツを検索できないことに注意してください。

（注） パイプライン処理された HTTP 要求パケットには複数の URI が含まれています。[HTTP URI] が選択されている場合、パイプライン処理されたHTTP要求パケットをルールエンジンが検出すると、そのパケット内のすべての URI でコンテンツ一致が検索されます。

HTTP Raw URI

正規化された要求URIフィールド内でコンテンツ一致を検索するには、このオプションを選択します。

このオプションと pcre キーワードの HTTP URI（U）オプションを一緒に使用して、同じコンテンツを検索できないことに注意してください。

（注） パイプライン処理された HTTP 要求パケットには複数の URI が含まれています。[HTTP URI] が選択されている場合、パイプライン処理されたHTTP要求パケットをルールエンジンが検出すると、そのパケット内のすべての URI でコンテンツ一致が検索されます。

HTTP メソッド（HTTP Method）

（URI で識別されるリソースに対して行う GET や POST などのアクションを特定する）要求メソッド フィールド内のコンテンツ一致を検索するには、このオプションを選択します。

HTTP ヘッダー（HTTP Header）

HTTP 要求内の（cookie を除く）正規化されたヘッダー フィールドでコンテンツ一致を検索するには、このオプションを選択します。また、HTTP Inspect プリプロセッサの [Inspect HTTP Responses] オプションが有効になっている場合は応答内でも検索されます。

このオプションと pcre キーワードのHTTP見出し（H）オプションを一緒に使用して、同じコンテンツを検索できないことに注意してください。

HTTP raw ヘッダー（HTTP Raw Header）

HTTP 要求内の（cookie を除く）raw ヘッダー フィールドでコンテンツ一致を検索するには、このオプションを選択します。また、HTTP Inspect プリプロセッサの [Inspect HTTP Responses] オプションが有効になっている場合は応答内でも検索されます。

このオプションと `pcre` キーワードの HTTP 未加工見出し（D）オプションを一緒に使用して、同じコンテンツを検索できないことに注意してください。

HTTP Cookie

正規化された HTTP クライアント要求見出し内で識別される cookie でコンテンツ一致を検索するには、このオプションを選択します。また、HTTP Inspect プリプロセッサの [HTTP 応答の検査（Inspect HTTP Responses）] オプションが有効になっている場合は応答 set-cookie データ内でも検索されます。システムは、メッセージ本文に含まれる cookie を本文の内容として扱うことに注意してください。

cookie 内だけで一致を検索するには、HTTP Inspect プリプロセッサの [HTTP Cookie の検査（Inspect HTTP Cookies）] オプションを有効にする必要があります。これを有効にしない場合、ルール エンジンは cookie を含む見出し全体を検索します。

次の点に注意してください。

- このオプションと `pcre` キーワードの HTTP cookie（C）オプションを一緒に使用して、同じコンテンツを検索することはできません。
- `Cookie:` ヘッダー名と `Set-Cookie:` ヘッダー名、ヘッダー行の先行スペース、およびヘッダー行の終わりを示す `CRLF` は cookie の一部としてではなく、ヘッダーの一部として検査されます。

HTTP Raw Cookie

未加工 HTTP クライアント要求見出し内で識別される cookie でコンテンツ一致を検索するには、このオプションを選択します。また、HTTP Inspect プリプロセッサの [HTTP 応答の検査（Inspect HTTP Responses）] オプションが有効になっている場合は応答 set-cookie データ内でも検索されます。システムは、メッセージ本文に含まれる cookie を本文の内容として扱うことに注意してください。

cookie 内だけで一致を検索するには、HTTP Inspect プリプロセッサの [HTTP Cookie の検査（Inspect HTTP Cookies）] オプションを有効にする必要があります。これを有効にしない場合、ルール エンジンは cookie を含む見出し全体を検索します。

次の点に注意してください。

- このオプションと `pcre` キーワードの HTTP 未加工 cookie（K）オプションを一緒に使用して同じコンテンツを検索することはできません。
- `Cookie:` ヘッダー名と `Set-Cookie:` ヘッダー名、ヘッダー行の先行スペース、およびヘッダー行の終わりを示す `CRLF` は cookie の一部としてではなく、ヘッダーの一部として検査されます。

HTTP クライアントボディ（HTTP Client Body）

HTTP クライアント要求内のメッセージ本文でコンテンツ一致を検索するには、このオプションを選択します。

このオプションが機能するためには、HTTP Inspect プリプロセッサの [HTTP クライアント ボディの抽出の深さ（HTTP Client Body Extraction Depth）] オプションで 0 ～ 65535 の値を指定する必要があることに注意してください。

HTTP ステータス コード（HTTP Status Code）

HTTP 応答内の 3 桁のステータス コードでコンテンツ一致を検索するには、このオプションを選択します。

このオプションで一致が返されるようにするには、HTTP Inspect プリプロセッサの [HTTP 応答の検査（Inspect HTTP Responses）] オプションを有効にする必要があります。

HTTP ステータスメッセージ（HTTP Status Message）

HTTP 応答のステータス コードに付加されるテキスト記述の中でコンテンツ一致を検索するには、このオプションを選択します。

このオプションで一致が返されるようにするには、HTTP Inspect プリプロセッサの [HTTP 応答の検査（Inspect HTTP Responses）] オプションを有効にする必要があります。

関連トピック

PCRE 修飾子のオプション （57 ページ）
サーバーレベルの HTTP 正規化オプション

概要：content キーワードによる高速パターン マッチ機能

（注） これらのオプションは、`protected_content` キーワードの設定では**サポートされません**。

高速パターン マッチ機能は、パケットをルール エンジンに渡す前に、評価するルールをすばやく決定します。この初期決定により、パケット評価で使用されるルール数が大幅に減るため、パフォーマンスが向上します。

デフォルトで、高速パターンマッチ機能は、ルールで指定された最長のコンテンツをパケットで検索します。これは、不必要なルール評価をできるだけ減らすためです。次の例のようなルール フラグメントがあるとします。

```
alert tcp any any -> any 80 (msg:"Exploit"; content:"GET";
http_method; nocase; content:"/exploit.cgi"; http_uri;
nocase;)
```

ほとんどすべての HTTP クライアント要求にはコンテンツ `GET` が含まれていますが、コンテンツ `/exploit.cgi` を含む要求は稀です。`GET` を高速パターン コンテンツとして使用した場合、ルール エンジンはほとんどのケースでこのルールを評価し、一致はほとんど検出されないで

しょう。しかし、/exploit.cgi を使用するとほとんどのクライアントの GET 要求は評価されないため、パフォーマンスが向上します。

指定されたコンテンツが高速パターン マッチ機能で検出された場合にのみ、ルール エンジンはパケットをルールに照らして評価します。たとえば、ルール内の 1 つの content キーワードでコンテンツ short を指定し、別のキーワードで longer、さらに 3 番目のキーワードで longest を指定した場合、高速パターン マッチ機能はコンテンツ longest を使用し、ルール エンジンがペイロード内で longest を検出した場合にのみ、ルールが評価されます。

content キーワードの高速パターン マッチ機能の引数

高速パターン マッチ機能を使用（Use Fast Pattern Matcher）

使用する高速パターンマッチ機能の短い検索パターンを指定するには、このオプションを使用します。理論的には、指定したパターンの方が最長パターンよりもパケット内で見つかる可能性が低いため、より的を絞って対象のエクスプロイトを識別できます。

Use Fast Pattern Matcher と他のオプションを同じ content キーワード内で選択する場合は、次の制限事項に注意してください。

- ルールごとに 1 回だけ、**Use Fast Pattern Matcher** を指定できます。
- **Use Fast Pattern Matcher** と **Not** を組み合わせて選択した場合は、**Distance**、**Within**、**Offset**、および **Depth** を使用できません。
- Use Fast Pattern Matcher を、次のいずれかの HTTP フィールド オプションと組み合わせて選択することはできません。

  [HTTP Raw URI]、[HTTP raw ヘッダー（HTTP Raw Header）]、[HTTP raw クッキー（HTTP Raw Cookie）]、[HTTP クッキー（HTTP Cookie）]、[HTTP メソッド（HTTP Method）]、[HTTP ステータス メッセージ（HTTP Status Message）]、または [HTTP ステータス コード（HTTP Status Code）]

  ただし、次のいずれかの正規化フィールドを検索するために高速パターンマッチ機能を使用する content キーワードでは、上記のオプションを含めることができます。

  [HTTP URI]、[HTTP ヘッダー（HTTP Header）]、または [HTTP クライアント ボディ（HTTP Client Body）]

  たとえば、[HTTP Cookie]、[HTTP Header]、および [Use Fast Pattern Matcher] を選択した場合、ルール エンジンは HTTP cookie と HTTP ヘッダーの両方でコンテンツを検索しますが、高速パターン マッチ機能は HTTP cookie ではなく、HTTP ヘッダーにのみ適用されます。

  未加工 HTTP フィールド オプション（[HTTP Raw URI]、[HTTP raw ヘッダー（HTTP Raw Header）]、または [HTTP raw クッキー（HTTP Raw Cookie）]）と、それぞれに対応する正規化されたオプション（[HTTP URI]、[HTTP ヘッダー（HTTP Header）]、または [HTTP クッキー（HTTP Cookie）]）を同じ content キーワード内で一緒に使用できないことに注意してください。

制限付きオプションと制限なしオプションを併用した場合、高速パターンマッチ機能は、指定された制限なしフィールドのみを検索することで、ルールエンジンにパケットを渡して（制限付きフィールドの評価を含む）完全な評価を行うべきかどうかを検査します。

- オプションで、**Use Fast Pattern Matcher** を選択した場合には **Fast Pattern Matcher Only** または **Fast Pattern Matcher Offset and Length** を選択することもできますが、この両方は選択できません。
- Base64 データの検査時には高速パターン マッチ機能を使用できません。

高速パターンマッチ機能のみ（Fast Pattern Matcher Only）

このオプションを使用すると、`content` キーワードをルール オプションとしてではなく、高速パターン マッチ機能オプションとしてのみ使用できます。指定したコンテンツをルール エンジンで評価する必要がない場合、このオプションを使ってリソースを節約できます。たとえば、ペイロード内のいずれかの場所にコンテンツ `12345` が存在することだけを必要とするルールがあるとします。高速パターンマッチ機能でパターンが検出された場合に、ルール内の追加のキーワードに照らしてパケットを評価できます。パターン `12345` が含まれているかどうかを判断するために、ルール エンジンがパケットを再評価する必要はありません。

指定されたコンテンツに関連する他の条件がルールに含まれている場合は、このオプションを使用しないでください。たとえば、別のルール条件で `abcd` が `1234` の前に出現するかどうかを判断する場合には、このオプションを使ってコンテンツ `1234` を検索しないでください。**Fast Pattern Matcher Only** を指定すると、指定されたコンテンツがルール エンジンによって検索されないため、このケースではルール エンジンが相対的な位置を判断できません。

このオプションを使用するときには、次の条件に注意してください。

- 指定されたコンテンツは位置に依存しない、つまり、ペイロードのどこにでも出現する可能性があるため、位置オプション（[距離（Distance）]、[次の範囲内（Within）]、[オフセット（Offset）]、[奥行き（Depth）]、[高速パターン マッチ機能オフセットおよび長さ（Fast Pattern Matcher Offset and Length）]）を使用することはできません。
- このオプションを **Not** と組み合わせて使用することはできません。
- このオプションを **Fast Pattern Matcher Offset and Length** と組み合わせて使用することはできません。
- 大文字/小文字を区別しない方法ですべてのパターンが高速パターン マッチ機能に挿入されるため、指定したコンテンツは「大文字/小文字の区別なし」として扱われます。これは自動的に処理されるため、このオプションの選択時に **Case Insensitive** を選択する必要はありません。
- **Fast Pattern Matcher Only** オプションを使用する `content` キーワードの直後に、現在の検索位置を基準にして検索位置を設定する次のキーワードを続けないようにしてください。
  - `isdataat`
  - `pcre`
  - `content`（[距離（Distance）] または [次の範囲内（Within）] が選択されている場合）

- content（[HTTP URI] が選択されている場合）
- asn1
- byte_jump
- byte_test
- byte_math
- byte_extract
- base64_decode

高速パターンマッチ機能オフセットおよび長さ（Fast Pattern Matcher Offset and Length）

[高速パターンマッチ機能オフセットおよび長さ（Fast Pattern Matcher Offset and Length）] オプションを使用すると、検索するコンテンツの一部分を指定できます。これにより、パターンが非常に長く、ルールの一致の可能性を判断するのにパターンの一部分だけで十分な場合に、メモリ消費を抑えることができます。高速パターンマッチ機能によってルールが選択されたときに、パターン全体がルールに照らして評価されます。

次の構文に従い、検索を開始する位置（オフセット）およびコンテンツ内をどれほど検索するか（長さ）をバイト単位で指定することにより、高速パターンマッチ機能で使用する部分を決定します。

```
offset,length
```

たとえば、次のコンテンツに対して

```
1234567
```

次のようにオフセットと長さのバイト数を指定した場合、

```
1,5
```

高速パターン マッチ機能はコンテンツ `23456` のみを検索します。

このオプションを [Fast Pattern Matcher Only] と一緒に使用できないことに注意してください。

関連トピック

概要：HTTP content および protected_content キーワードの引数 （34 ページ）
base64_decode キーワードと base64_data キーワード （131 ページ）

replace キーワード

インライン導入で `replace` キーワードを使用すると、指定したコンテンツ、または Cisco SSL アプライアンスによって検出された SSL トラフィック内のコンテンツを置き換えることができます。

`replace` キーワードを使用するには、`content` キーワードを使って特定の文字列を検索するカスタムの標準テキスト ルールを作成します。その後、`replace` キーワードを使用して、コンテン

ツを置き換える文字列を指定します。置換値とコンテンツ値は同じ長さである必要があります。

（注） protected_content キーワード内でハッシュされたコンテンツを置き換えるために replace キーワードを使用することは**できません**。

オプションで、以前のソフトウェアバージョンとの下位互換性を維持するために、置換文字列を引用符で囲むことができます。引用符を含めない場合は、それらが自動的にルールに追加されるため、構文的に正しいルールになります。置換テキストの一部として先行引用符または後続引用符を含めるには、次の例に示すように、バックスラッシュを使ってエスケープする必要があります。

```
"replacement text plus \"quotation\" marks""
```

1 つのルール内に複数の replace キーワードを含めることができますが、content キーワードごとに 1 つずつしか含めることができません。ルールによって検出されたコンテンツの最初のインスタンスだけが置き換えられます。

次に、replace キーワードの使用例を示します。

- エクスプロイトを含んでいる着信パケットをシステムが検出した場合、有害な文字列を無害な文字列に置き換えることができます。このテクニックは、有害なパケットを単に破棄するよりも効果的である場合があります。破棄されたパケットを攻撃者が単に再送信し続け、やがてネットワーク防御を通り抜けるか、ネットワークを氾濫させるという攻撃シナリオがあります。パケットを破棄する代わりに別の文字列に置換することで、脆弱ではないターゲットに対して攻撃が実行されたと攻撃者に思い込ませることができます。
- （たとえば Web サーバーの）脆弱なバージョンが稼働しているかどうかを調べる偵察攻撃が懸念される場合は、発信パケットを検出して、バナーを独自のテキストに置換できます。

（注） 置換ルールを使用するインライン侵入ポリシー内でルール状態が[イベントを生成する（Generate Events）]に設定されていることを確認してください。ルールを[ドロップしてイベントを生成する（Drop and Generate Events）]に設定した場合はパケットが破棄され、コンテンツが置き換えられません。

文字列置換プロセスでは、宛先ホストがエラーなしでパケットを受信できるように、パケットチェックサムがシステムによって自動的に更新されます。

replace キーワードは、HTTP 要求メッセージの content キーワード オプションと組み合わせて使用できないことに注意してください。

関連トピック

content キーワードと protected_content キーワード （28 ページ）
概要：HTTP content および protected_content キーワードの引数 （34 ページ）

byte_jump キーワード

byte_jump キーワードは、指定されたバイト セグメントで定義されるバイト数を計算し、指定したオプションに応じて、指定されたバイト セグメントの末尾から、パケット ペイロードの先頭または末尾から、あるいは最後のコンテンツ一致に対して相対的なポイントから順方向に、パケット内でそのバイト数だけスキップします。パケットの特定のバイトセグメントが、パケット内の可変データに含まれるバイト数を示す場合には、これが役立ちます。

次の表では、byte_jump キーワードで必要な引数を説明します。

表 *10 : byte_jump* の必須引数

引数	説明
Bytes	パケットから抽出するバイト数。 DCE/RPC を指定せずに使用する場合、許可される値は 0 ～ 10 ですが、次の制限があります。 • From End 引数とともに使用すると、バイト数は 0 になることがあります。Bytes が 0 の場合、抽出された値は 0 です。 • 1、2、または 4 以外のバイト数を指定する場合は、番号タイプ（16 進数、8 進数、または 10 進数）を指定する必要があります。 DCE/RPC とともに使用する場合、許可される値は 1、2、および 4 です。
Offset	ペイロード内で処理を開始するバイト数。offset カウンタはバイト 0 から始まるため、パケット ペイロードの先頭、または最後に見つかったコンテンツ一致から順方向にジャンプさせるバイト数から 1 を差し引いて offset 値を計算してください。 -65535 ～ 65535 バイトを指定できます。 また、既存の byte_extract 変数または byte_math 結果を使用してこの引数の値を指定することもできます。

次の表で説明するオプションを使用すると、必須の引数に指定された値をシステムがどのように解釈するかを定義できます。

表 *11 : byte_jump* の追加のオプション引数

引数	説明
Relative	最後に見つかったコンテンツ一致で検出された最後のパターンを基準にしてオフセットを計算します。
Align	変換されたバイト数を、次の 32 ビット境界に切り上げます。

引数	説明
Multiplier	ルール エンジンで最終的な byte_jump 値を算出するために、パケットから得られた byte_jump 値に掛ける値を示します。 つまり、ルール エンジンは、指定されたバイト セグメントで定義されるバイト数だけスキップする代わりに、Multiplier 引数で指定される整数を乗算したバイト数だけスキップします。
Post Jump Offset	他の byte_jump 引数を適用した後に、順方向または逆方向にスキップするバイト数（-65535 ～ 65535）。正の値は順方向にスキップし、負の値は逆方向にスキップします。無効にするには、フィールドを空白のままにするか、0 を入力します。 **DCE/RPC** 引数を選択すると、一部の byte_jump 引数が適用されないことに注意してください。
From Beginning	ルール エンジンが、パケット内の現在の位置からではなく、パケット ペイロードの先頭からペイロード内の指定されたバイト数をスキップする必要があることを示します。
From End	ジャンプは、バッファの最後のバイトのすぐ後のバイトから実行されます。
Bitmask	AND 演算子を使用して、指定した 16 進数のビットマスクを、Bytes 引数から抽出したバイトに適用します。 ビットマスクは 1 ー 4 バイトです。 結果は、マスク内の末尾のゼロの数と等しい数のビット分だけ右にシフトされます。

DCE/RPC、**Endian**、または **Number Type** のうち 1 つだけを指定できます。

バイト数を byte_extract キーワードでどのように計算するか定義するには、次の表の中から引数を選択します。バイト順引数を選択しなかった場合、ルール エンジンはビッグ エンディアンのバイト順を使用します。

表 *12 : byte_jump* のバイト順引数

引数	説明
Big Endian	デフォルトのネットワーク バイト順であるビッグ エンディアン バイト順でデータを処理します。
Little Endian	リトル エンディアン バイト順でデータを処理します。

引数	説明
DCE/RPC	DCE/RPC プリプロセッサで処理されるトラフィック用に `byte_jump` キーワードを指定します。 DCE/RPC プリプロセッサがビッグ エンディアンまたはリトル エンディアン バイト順を決定します。**Number Type** 引数と **Endian** 引数は適用されません。 この引数を有効にした場合は、他の特定の DCE/RPC キーワードと組み合わせて `byte_jump` を使用することもできます。

次の表に示すいずれか1つの引数を使用して、パケット内のストリングデータをシステムがどのように表示するかを定義します。

表 *13*：番号タイプ引数

引数	説明
Hexadecimal String	変換後のストリング データを 16 進形式で表現します。
Decimal String	変換後のストリング データを 10 進形式で表現します。
Octal String	変換後のストリング データを 8 進形式で表現します。

たとえば、次のような値を `byte_jump` に設定した場合、

- Bytes = 4
- Offset = 12
- Relative enabled
- Align enabled

ルール エンジンは、最後に見つかったコンテンツ一致から 13 バイト後に出現する 4 つのバイトで記述される数値を計算して、そのバイト数だけパケット内を順方向にスキップします。たとえば、ある特定のパケット内で計算される 4 つのバイトが `00 00 00 1F` である場合、ルール エンジンはこれを 31 に変換します。`align` が指定されている（次の 32 ビット境界まで移動するようエンジンに指示する）ため、ルール エンジンはパケット内を 32 バイト先までスキップします。

あるいは、次のような値を `byte_jump` に設定した場合、

- Bytes = 4
- Offset = 12
- From Beginning enabled
- Multiplier = 2

ルール エンジンは、パケットの先頭から 13 バイト後に出現する 4 つのバイトで記述される数値を計算します。その後、その数値に 2 を掛けてスキップする総バイト数を計算します。たとえば、ある特定のパケット内で計算される 4 つのバイトが `00 00 00 1F` である場合、ルール エンジンはこれを 31 に変換し、それに 2 を掛けて 62 にします。[From Beginning] が有効になっているため、ルール エンジンはパケット内の最初の 63 バイトをスキップします。

関連トピック

byte_extract キーワード （48 ページ）
DCE/RPC キーワード （86 ページ）

byte_test キーワード

`byte_test` キーワードは、指定されたバイト セグメントを Value 引数およびその演算子に対してテストします。

次の表に、`byte_test` キーワードで必要な引数を説明します。

表 *14 : byte_test* の必須引数

引数	説明
Bytes	パケットから計算するバイト数。 DCE/RPC を指定せずに使用する場合、許可される値は 1 ～ 10 です。ただし、1、2、または 4 以外のバイト数を指定する場合は、番号タイプ（16 進数、8 進数、または 10 進数）を指定する必要があります。 DCE/RPC とともに使用する場合、許可される値は 1、2、および 4 です。
値	テストする値（演算子を含む）。 サポート対象演算子：<、>、=、!、&、^、!>、!<、!=、!&、または !^。 たとえば `!1024` と指定した場合、`byte_test` は指定された数値を変換し、それが 1024 と等しくなければイベントが生成されます（他のすべてのキーワード パラメータが一致する場合）。 「!」と「!=」は等価であることに注意してください。 また、既存の `byte_extract` 変数または `byte_math` 結果を使用してこの引数の値を指定することもできます。
Offset	ペイロード内で処理を開始するバイト数。`offset` カウンタはバイト 0 から始まるため、パケット ペイロードの先頭、または最後に見つかったコンテンツ一致から順方向にカウントするバイト数から 1 を差し引いて `offset` 値を計算してください。 既存の `byte_extract` 変数または `byte_math` result 変数を使用して、この引数の値を指定することができます。

次の表に示す引数を使用すると、システムで byte_test 引数がどのように使用されるかをさらに定義できます。

表 *15 : byte_test* の追加のオプション引数

引数	説明
Bitmask	AND 演算子を使用して、指定した 16 進数のビットマスクを、Bytes 引数から抽出したバイトに適用します。 ビットマスクは 1 ー 4 バイトです。 結果は、マスク内の末尾のゼロの数と等しい数のビット分だけ右にシフトされます。
Relative	最後に見つかったパターン一致を基準にしてオフセットを計算します。

DCE/RPC、**Endian**、または **Number Type** のうち 1 つだけを指定できます。

検査対象となるバイトを byte_test キーワードでどのように計算するか定義するには、次の表の中から引数を選択します。バイト順引数を選択しなかった場合、ルール エンジンはビッグ エンディアンのバイト順を使用します。

表 *16 : byte_test* のバイト順引数

引数	説明
Big Endian	デフォルトのネットワーク バイト順であるビッグ エンディアン バイト順でデータを処理します。
Little Endian	リトル エンディアン バイト順でデータを処理します。
DCE/RPC	DCE/RPC プリプロセッサで処理されるトラフィック用に byte_test キーワードを指定します。 DCE/RPC プリプロセッサがビッグ エンディアンまたはリトル エンディアン バイト順を決定します。**Number Type** 引数と **Endian** 引数は適用されません。 この引数を有効にした場合は、他の特定の DCE/RPC キーワードと組み合わせて byte_test を使用することもできます。

次の表に示すいずれか1つの引数を使用して、パケット内のストリング データをシステムがどのように表示するかを定義できます。

表 *17 : byte-test* の番号タイプ引数

引数	説明
Hexadecimal String	変換後のストリング データを 16 進形式で表現します。
Decimal String	変換後のストリング データを 10 進形式で表現します。

引数	説明
Octal String	変換後のストリング データを 8 進形式で表現します。

たとえば、次のような値を byte_test に指定した場合、

- Bytes = 4
- Operator and Value > 128
- Offset = 8
- Relative enabled

ルールエンジンは、最後に見つかったコンテンツ一致から（それを基準にして）9バイト後に出現する 4 つのバイトで記述される数値を計算し、その計算値が 128 バイトを超えた場合に、ルールがトリガーとして使用されます。

関連トピック

byte_extract キーワード （48 ページ）
DCE/RPC キーワード （86 ページ）

byte_extract キーワード

byte_extract キーワードを使用すると、指定したバイト数をパケットから変数の中に読み込むことができます。後で、その変数を、同じルール内で他の検出キーワードの特定の引数の値として使用できます。

たとえば、パケット データに含まれるバイト数が特定のバイト セグメントで記述されている場合、パケットからデータ サイズを抽出するには、これが役立ちます。たとえば、特定のバイト セグメントにおいて、後続データが 4 バイト構成であると記述されている場合、データ サイズ 4 バイトを抽出して変数値として使用できます。

byte_extract を使用するとき、1 つのルール内で最大 2 つの異なる変数を同時に作成できます。byte_extract 変数を何回でも再定義できます。同じ変数名と別の変数定義を使って新しい byte_extract キーワードを入力した場合、その前の変数定義がオーバーライドされます。

次の表に、byte_extract キーワードで必要な引数について説明します。

表 *18 : byte_extract* の必須引数

引数	説明
Bytes to Extract	パケットから抽出するバイト数。 1、2、または 4 以外のバイト数を指定する場合は、番号タイプ（16 進数、8 進数、または 10 進数）を指定する必要があります。

引数	説明
Offset	ペイロード内でデータの抽出を開始するバイト数。-65535 ～ 65535 バイトを指定できます。オフセットカウンタはバイト 0 から始まるため、順方向に数えるバイト数から 1 を差し引いてオフセット値を計算してください。たとえば、順方向に 8 バイト数えるには 7 を指定します。ルール エンジンは、パケット ペイロードの先頭から（**Relative** も一緒に指定した場合は最後に見つかったコンテンツ一致の後から）順方向に数えます。負の数は、**Relative** も指定した場合にのみ指定できます。 既存の `byte_math` の結果を使用して、この引数の値を指定することもできます。
Variable Name	他の検出キーワードの引数で使用する変数名。英数字の文字列を指定できます（ただし文字で始まる必要があります）。

抽出対象のデータを見つける方法をさらに詳しく定義するには、次の表に示す引数を使用できます。

表 *19 : byte_extract* の追加のオプション引数

引数	説明
Multiplier	パケットから抽出された値の乗数。0 ～ 65535 を指定できます。乗数を指定しない場合のデフォルト値は 1 です。
Align	抽出された値を最も近い 2 バイトまたは 4 バイト境界に切り上げます。**Multiplier** も一緒に選択した場合、システムはこの調整の前に乗数を適用します。
Relative	ペイロードの先頭ではなく、最後に見つかったコンテンツ一致の末尾を基準にして **Offset** を計算します。
Bitmask	AND 演算子を使用して、指定した 16 進数のビットマスクを、Bytes to Extract 引数から抽出したバイトに適用します。 ビットマスクは 1 ー 4 バイトです。 結果は、マスク内の末尾のゼロの数と等しい数のビット分だけ右にシフトされます。

DCE/RPC、**Endian**、または **Number Type** のうち 1 つだけを指定できます。

検査対象となるバイトを `byte_extract` キーワードでどのように計算するか定義するには、次の表の中から引数を選択できます。バイト順引数を選択しなかった場合、ルール エンジンはビッグ エンディアンのバイト順を使用します。

表 *20 : byte_extract* のバイト順引数

引数	説明
Big Endian	デフォルトのネットワーク バイト順であるビッグ エンディアン バイト順でデータを処理します。
Little Endian	リトル エンディアン バイト順でデータを処理します。
DCE/RPC	DCE/RPC プリプロセッサで処理されるトラフィック用に `byte_extract` キーワードを指定します。 DCE/RPC プリプロセッサがビッグ エンディアンまたはリトル エンディアン バイト順を決定します。**Number Type** 引数と **Endian** 引数は適用されません。 この引数を有効にした場合は、他の特定の DCE/RPC キーワードと組み合わせて `byte_extract` を使用することもできます。

データを読み取るときの数値タイプを ASCII 文字列として指定できます。パケット内のストリング データをシステムがどのように認識するかを定義するには、次の表のいずれかの引数を選択できます。

表 *21 : byte_extract* の番号タイプ引数

引数	説明
Hexadecimal String	抽出されたストリング データを 16 進形式で読み取ります。
Decimal String	抽出されたストリング データを 10 進形式で読み取ります。
Octal String	抽出されたストリング データを 8 進形式で読み取ります。

たとえば、`byte_extract` の値を次のように指定した場合、

- Bytes to Extract = 4
- Variable Name = var
- Offset = 8
- Relative = enabled

ルール エンジンは、最後に見つかったコンテンツ一致から（それを基準にして）9 バイト後に出現する、4 バイトで表現される数値を `var` という名前の変数の中に読み込みます。後でこの変数を、特定のキーワード引数の値としてルール内で指定できます。

`byte_extract` キーワードで定義した変数を指定できるキーワード引数を、次の表に列挙します。

表 *22 : byte_extract* 変数を使用できる引数

キーワード	引数
content	Depth、Offset、Distance、Within
byte_jump	Offset
byte_test	Offset、Value
byte_math	RValue、Offset
isdataat	Offset

関連トピック

DCE/RPC プリプロセッサ
DCE/RPC キーワード （86 ページ）
基本コンテンツおよび protected_content キーワードの引数 （30 ページ）
byte_jump キーワード （43 ページ）
byte_test キーワード （46 ページ）
パケット特性 （111 ページ）

byte_math キーワード

byte_math キーワードは、抽出された値と指定された値または既存の変数の算術演算を実行し、その結果を新しい結果変数に格納します。結果の変数は、他のキーワードの引数として使用することができます。

ルール内で複数の byte_math キーワードを使用して、複数の byte_math 操作を実行できます。

次の表で、byte_math キーワードで必要な引数について説明します。

表 *23 : byte_math* の必須引数

引数	説明
Bytes	パケットから計算するバイト数。 DCE/RPC を指定せずに使用する場合、許可される値は 1 ～ 10 です。 • 演算子が +、-、*、または / の場合、バイト数は 1 ー 10 になります。 • 演算子が << または >> の場合、バイト数は 1 ー 4 になります。 • 1、2、または4以外のバイト数を指定する場合は、番号タイプ（16進数、8 進数、または 10 進数）を指定する必要があります。 DCE/RPC とともに使用する場合、許可される値は 1、2、および 4 です。

引数	説明
Offset	ペイロード内で処理を開始するバイト数。`offset` カウンタはバイト 0 から始まるため、パケットペイロードの先頭、または最後に見つかったコンテンツ一致（Relative を指定した場合）から順方向にジャンプさせるバイト数から 1 を差し引いて `offset` 値を計算してください。 -65535 ～ 65535 バイトを指定できます。 ここでは、`byte_extract` 変数を指定することもできます。
演算子	+、-、*、/、<<、または >>
RValue	演算子に続く値。これは、符号なし整数または `byte_extract` から渡される変数です。
Result Variable	`byte_math` の計算結果が格納される変数の名前。この変数は、他のキーワードの引数として使用することができます。 この値は符号なし整数として格納されます。 この変数名には次の条件があります。 • 英数字を使用する必要がある • 先頭を数字にすることはできない • Microsoft のファイル名/変数名の規則でサポートされている特殊文字を含めることができる • 特殊文字のみの名前にすることはできない

次の表で説明するオプションを使用すると、必須の引数に指定された値をシステムがどのように解釈するかを定義できます。

表 *24 : byte_math* の追加のオプション引数

引数	説明
Relative	ペイロードの先頭ではなく、最後に見つかったコンテンツ一致で検出された最後のパターンを基準にしてオフセットを計算します。
Bitmask	AND 演算子を使用して、指定した 16 進数のビットマスクを、Bytes 引数から抽出したバイトに適用します。 ビットマスクは 1 ー 4 バイトです。 結果は、マスク内の末尾のゼロの数と等しい数のビット分だけ右にシフトされます。

DCE/RPC、**Endian**、または **Number Type** のうち 1 つだけを指定できます。

バイト数を byte_math キーワードでどのように計算するか定義するには、次の表の中から引数を選択します。バイト順引数を選択しなかった場合、ルール エンジンはビッグ エンディアンのバイト順を使用します。

表 *25 : byte_math* のバイト順引数

引数	説明
Big Endian	デフォルトのネットワーク バイト順であるビッグ エンディアン バイト順でデータを処理します。
Little Endian	リトル エンディアン バイト順でデータを処理します。
DCE/RPC	DCE/RPC プリプロセッサで処理されるトラフィック用に byte_math キーワードを指定します。 DCE/RPC プリプロセッサがビッグ エンディアンまたはリトル エンディアン バイト順を決定します。**Number Type** 引数と **Endian** 引数は適用されません。 この引数を有効にした場合は、他の特定の DCE/RPC キーワードと組み合わせて byte_math を使用することもできます。

次の表に示すいずれか1つの引数を使用して、パケット内のストリング データをシステムがどのように表示するかを定義します。

表 *26 :* 番号タイプ引数

引数	説明
Hexadecimal String	ストリング データを 16 進形式で表現します。
Decimal String	ストリング データを 10 進形式で表現します。
Octal String	ストリング データを 8 進形式で表現します。

たとえば、次のような値を byte_math に設定した場合、

- Bytes = 2
- Offset = 0
- Operator = *
- RValue = height
- Result Variable = area

ルール エンジンは、パケット内の最初の 2 バイトに記述された番号を抽出し、RValue（既存の変数 height を使用）を乗じて新しい変数 area を作成します。

表 *27 : byte_math* 変数を使用できる引数

キーワード	引数
byte_jump	Offset
byte_test	Offset、Value
byte_extract	Offset
isdataat	Offset

概要：pcre キーワード

pcre キーワードを使用すると、指定されたコンテンツをパケット ペイロード内で検査するために Perl 互換正規表現（PCRE）を使用できます。PCRE を使用すると、同じ内容のわずかなバリエーションにそれぞれ一致する複数のルールを作成する手間が省けます。

正規表現は、さまざまな方法で表現されることのあるコンテンツを検索する場合に役立ちます。パケットのペイロード内でコンテンツを検索するときには、コンテンツがさまざまな属性を持つ可能性があることを考慮すべき場合があります。

侵入ルールで使われる正規表現構文は完全な正規表現ライブラリのサブセットであり、完全なライブラリ内のコマンドで使用される構文とはいくつかの点で異なることに注意してください。侵入ルール エディタを使用して pcre キーワードを追加するときには、次の形式で完全な値を入力します。

```
!/pcre/ ismxAEGRBUIPHDMCKSY
```

引数の説明

- 「!」は否定オプションです（正規表現に**一致しない**パターンを照合する場合に使用します）。
- /pcre/ は Perl 互換正規表現です。
- ismxAEGRBUIPHDMCKSY は修飾子オプションの任意の組み合わせです。

また、次の表に示す文字をエスケープする必要があることに注意してください。これにより、パケット ペイロード内で特定のコンテンツを検索するために PCRE でこれらの文字を使用した場合、ルール エンジンがそれを正しく解釈するようになります。

表 *28 :* エスケープする *PCRE* 文字

エスケープする必要のある文字	バックスラッシュを使用した場合	16進コードを使用した場合
#（ナンバー記号）	\#	\x23
;（セミコロン）	\;	\x3B

エスケープする必要のある文字	バックスラッシュを使用した場合	16進コードを使用した場合
|（縦棒）	|	\x7C
：（コロン）	\:	\x3A

m?regex? を使用することもできます。ここで、? は「/」以外のデリミタです。正規表現内でスラッシュと一致させる必要があり、バックスラッシュを使ってそれをエスケープしたくない場合には、これを使用できます。たとえば、m?regex? ismxAEGRBUIPHDMCKSY を使用できます。ここで regex は Perl 互換正規表現、ismxAEGRBUIPHDMCKSY は修飾子オプションの任意の組み合わせです。

ヒント 必要に応じて、Perl 互換正規表現を引用符で囲むことができます。たとえば、pcre_expression は "pcre_expression" となります。引用符が任意ではなく必須であった旧バージョンに慣れている経験豊富なユーザのために、引用符を使用するオプションが提供されています。保存後のルールを侵入ルール エディタで表示すると、引用符が表示されません。

PCRE の構文

pcre キーワードでは、標準の Perl 互換正規表現（PCRE）構文を使用できます。以下の項では、この構文について説明します。

ヒント ここでは PCRE で使用可能な基本的な構文について説明しますが、Perl および PCRE 専用のオンライン リファレンスやブックで、さらに詳しい情報を参照することもできます。

メタ文字

メタ文字は正規表現内で特別な意味を持つリテラル文字です。メタ文字を正規表現内で使用するときは、その前にバックスラッシュを付けて「エスケープする」必要があります。

次の表に、PCRE で使用可能なメタ文字について説明し、それぞれの例を示します。

表 *29 : PCRE* メタ文字

メタ文字	説明	例
。	改行以外の任意の文字と一致します。修飾オプションとして s が使用されている場合は、改行文字も含まれます。	abc. は、abcd、abc1、abc# などと一致します。
*。	ある文字または式の 0 回以上の出現と一致します。	abc* は、abc、abcc、abccc、abcccccc などと一致します。

メタ文字	説明	例
?	ある文字または式の 0 回または 1 回の出現と一致します。	abc? は abc に一致します。
+	ある文字または式の 1 回以上の出現と一致します。	abc+ は、abc、abcc、abccc、abccccc などと一致します。
()	式をグループ化します。	(abc)+ は、abc、abcabc、abcabcabc などと一致します。
{}	ある文字または式の一致回数の限度を指定します。下限と上限を設定する場合には、下限と上限をカンマで区切ります。	a{4,6} は、aaaa、aaaaa、または aaaaaa と一致します。 (ab){2} は abab と一致します。
[]	文字クラスを定義できます。セットの中で記述される任意の文字または文字の組み合わせに一致します。	[abc123] は、a または b または c などと一致します。
^	文字列の先頭でコンテンツを照合します。また、文字クラスの中で否定としても使用されます。	^in は、info 内の「in」と一致しますが、bin では一致しません。[^a] は、a を含まない任意の文字列と一致します。
$	文字列の末尾でコンテンツを照合します。	ce$ は、announce 内の「ce」と一致しますが、cent では一致しません。
|	OR 式を示します。	(MAILTO|HELP) は、MAILTO または HELP と一致します。
\	メタ文字を実際の文字として使用できます。また、事前定義された文字クラスを指定するためにも使われます。	\. はピリオドと一致し、* はアスタリスクと一致し、\\ はバックスラッシュと一致します。\d は数字と一致し、\w は英数字と一致します。

文字クラス

文字クラスには、英字、数字、英数字、および空白文字があります。大カッコで囲んで独自の文字クラスを作成できます。また、事前定義のクラスをさまざまな文字タイプのショートカットとして使用することもできます。追加の修飾子なしで文字クラスを使用すると、1 つの文字クラスは 1 桁または 1 文字に一致します。

次の表に、PCRE で使用できる事前定義の文字クラスの説明と例を示します。

表 *30 : PCRE* 文字クラス

文字クラス	説明	文字クラスの定義
\d	数字（桁）と一致します。	[0-9]
\D	数字以外の任意の文字と一致します。	[^0-9]

文字クラス	説明	文字クラスの定義
\w	英数字（語）と一致します。	[a-zA-Z0-9_]
\W	英数字以外の任意の文字と一致します。	[^a-zA-Z0-9_]
\s	スペース、復帰、タブ、改行、および改ページを含む空白文字と一致します。	[\r\t\n\f]
\S	空白文字以外の任意の文字と一致します。	[^ \r\t\n\f]

PCRE 修飾子のオプション

pcre キーワードの値の中で正規表現構文を指定した後、修飾オプションを使用できます。これらの修飾子は、Perl、PCRE、および Snort 固有の処理機能を実行します。修飾子は、常に PCRE 値の末尾に、次の形式で出現します。

```
/pcre/ismxAEGRBUIPHDMCKSY
```

ここで、ismxAEGRBUPHMC には、次の表に示す任意の修飾オプションを含めることができます。

ヒント オプションで、正規表現と修飾オプションを引用符で囲むことができます（たとえば "/pcre/ismxAEGRBUIPHDMCKSY"）。引用符が任意ではなく必須であった旧バージョンに慣れている経験豊富なユーザのために、引用符を使用するオプションが提供されています。保存後のルールを侵入ルール エディタで表示すると、引用符が表示されません。

次の表に、Perl 処理機能を実行するために使用できるオプションを説明します。

表 *31 : Perl* 関連の正規表現後オプション

オプション	説明
i	正規表現で大文字と小文字を区別しないようにします。
s	ドット文字（.）は、改行または \n 文字を除くすべての文字を表します。オプションとして "s" を使用すると、これをオーバーライドして、改行文字を含むすべての文字をドット文字に一致させることができます。
m	デフォルトで、1 つの文字列は複数文字からなる単一行として扱われ、^ と $ は特定の文字列の先頭および末尾に一致します。オプションとして "m" を使用すると、^ および $ はバッファの先頭または末尾だけでなく、バッファ内の改行文字の直前または直後のコンテンツとも一致します。
x	エスケープされた（バックスラッシュが先行する）場合、および文字クラスに含まれる場合を除き、空白データ文字がパターン内に出現してもそれを無視します。

次の表に、正規表現の後ろに使用できる PCRE 修飾子の説明を示します。

表 *32 : PCRE* 関連の正規表現後オプション

オプション	説明
A	文字列の先頭でパターンが一致する必要があります（正規表現で ^ を使用した場合と同じ）。
E	対象の文字列の末尾でのみ一致するように $ を設定します。（E を伴わない $ は、それが改行である場合には最後の文字の直前とも一致しますが、他の改行文字の直前とは一致しません）。
G	デフォルトでは、* + と ? は「最長マッチ」を実行します。つまり、複数の一致が見つかった場合は最も長い一致が選択されます。G 文字を使用するとこの動作が変更され、常に最初の一致がこれらの文字で選択されます。ただし後ろに疑問符（?）が続く場合を除きます。たとえば、*? +? と ?? は G 修飾子を使った構造内で最長マッチを実行し、疑問符が付いていない *、+、または ? は最長マッチではありません。

次の表に、正規表現の後ろに使用できる Snort 固有の修飾子の説明を示します。

.

表 *33 : Snort* 固有の正規表現後の修飾子

オプション	説明
R	ルールエンジンで見つかった最後の一致の末尾を基準にして、一致するコンテンツを検索します。
B	プリプロセッサによってデコードされる前のデータ内のコンテンツを検索します（このオプションは、`content` または `protected_content` キーワードで `Raw Data` 引数を使用する場合と似ています）。
U	HTTP Inspect プリプロセッサによってデコードされた正規化済み HTTP 要求メッセージの URI 内のコンテンツを検索します。このオプションと `content` または `protected_content` キーワードの **HTTP URI** オプションを一緒に使用して、同じコンテンツを検索することはできません。 パイプライン処理された HTTP 要求パケットには複数の URI が含まれていることに注意してください。U オプションを含む PCRE 式を使用すると、ルールエンジンは、パイプライン処理された HTTP 要求パケット内の最初の URI でのみコンテンツ一致を検索します。パケット内のすべての URI を検索するには、**HTTP URI** を選択して `content` または `protected_content` キーワードを使用します。U オプションを含む PCRE 式を一緒に使用するかどうかは問いません。

オプション	説明
I	HTTP Inspect プリプロセッサによってデコードされた raw HTTP 要求メッセージの URI 内のコンテンツを検索します。このオプションと `content` または `protected_content` キーワードの **HTTP Raw URI** オプションを一緒に使用して、同じコンテンツを検索することはできません。
P	HTTP Inspect プリプロセッサによってデコードされた正規化済み HTTP 要求メッセージ本文の中でコンテンツを検索します。
H	HTTP Inspect プリプロセッサによってデコードされた HTTP 要求または応答メッセージの（cookie を除く）ヘッダー内のコンテンツを検索します。このオプションと `content` または `protected_content` キーワードの **HTTP Header** オプションを一緒に使用して、同じコンテンツを検索することはできません。
D	HTTP Inspect プリプロセッサによってデコードされた未加工の HTTP 要求または応答メッセージの（cookie を除く）ヘッダー内のコンテンツを検索します。このオプションと `content` または `protected_content` キーワードの **HTTP Raw Header** オプションを一緒に使用して、同じコンテンツを検索することはできません。
M	HTTP Inspect プリプロセッサによってデコードされた正規化済み HTTP 要求メッセージのメソッドフィールド内のコンテンツを検索します。メソッドフィールドは、URI で識別されるリソースに対して実行すべきアクション（GET、PUT、CONNECT など）を特定します。
C	HTTP Inspect プリプロセッサの [HTTP Cookie の検査（Inspect HTTP Cookies）] オプションが有効になっている場合は、HTTP 要求見出しの cookie 内の正規化済みコンテンツを検索します。さらに、プリプロセッサの [HTTP 応答の検査（Inspect HTTP Responses）] オプションが有効になっている場合は、HTTP 応答見出しの set-cookie 内も検索します。[HTTP Cookie の検査（Inspect HTTP Cookies）] が有効になっていない場合は、cookie または set-cookie データを含む見出し全体を検索します。 次の点に注意してください。 • メッセージ本文に含まれる cookie は、本文のコンテンツとして扱われます。 • このオプションと `content` または `protected_content` キーワードの **HTTP Cookie** オプションを一緒に使用して、同じコンテンツを検索することはできません。 • `Cookie:` 見出し名と `Set-Cookie:` 見出し名、見出し行の先行スペース、および見出し行の終わりを示す `CRLF` は cookie の一部としてではなく、見出しの一部として検査されます。

オプション	説明
K	HTTP Inspect プリプロセッサの [HTTP Cookie の検査（Inspect HTTP Cookies）] オプションが有効になっている場合は、HTTP 要求見出しの cookie 内の未加工コンテンツを検索します。さらに、プリプロセッサの [HTTP 応答の検査（Inspect HTTP Responses）] オプションが有効になっている場合は、HTTP 応答見出しの set-cookie 内も検索します。[HTTP Cookie の検査（Inspect HTTP Cookies）] が有効になっていない場合は、cookie または set-cookie データを含む見出し全体を検索します。 次の点に注意してください。 • メッセージ本文に含まれる cookie は、本文のコンテンツとして扱われます。 • このオプションと `content` または `protected_content` キーワードの **HTTP Raw Cookie** オプションを一緒に使用して、同じコンテンツを検索することはできません。 • `Cookie:` 見出し名と `Set-Cookie:` 見出し名、見出し行の先行スペース、および見出し行の終わりを示す `CRLF` は cookie の一部としてではなく、見出しの一部として検査されます。
S	HTTP 応答内の 3 桁のステータス コードを検索します。
Y	HTTP 応答内のステータス コードに付加されるテキスト記述を検索します。

（注） U オプションと R オプションを組み合わせて使用しないでください。パフォーマンスの問題が発生する可能性があります。また、他の HTTP コンテンツ オプション（I、P、H、D、M、C、K、S または Y）と組み合わせて U オプションを使用しないでください。

関連トピック

概要：HTTP content および protected_content キーワードの引数 （34 ページ）

PCRE のキーワード値の例

次に、`pcre` で入力できる値の例を示し、それぞれの例で何が一致するかを説明します。

- **`/feedback[(\d{0,1})]?\.cgi/U`**

この例では、URI データにのみ配置された、`feedback` の後に 0 個または 1 個の数字、さらに `.cgi` が続くインスタンスをパケット ペイロード内で検索します。

この例は以下のものと一致します。

- `feedback.cgi`

- `feedback1.cgi`
- `feedback2.cgi`
- `feedback3.cgi`

この例は、以下のものとは**一致しません**。

- `feedbacka.cgi`
- `feedback11.cgi`
- `feedback21.cgi`
- `feedbackzb.cgi`

- **`/^ez(\w{3,5})\.cgi/iU`**

この例では、先頭の `ez` の後に 3 ～ 5 文字の単語、さらに `.cgi` が続く文字列をパケット ペイロード内で検索します。この検索では大文字と小文字を区別せず、URI データだけを検索します。

この例は以下のものと一致します。

- `EZBoard.cgi`
- `ezman.cgi`
- `ezadmin.cgi`
- `EZAdmin.cgi`

この例は、以下のものとは**一致しません**。

- `ezez.cgi`
- `fez.cgi`
- `abcezboard.cgi`
- `ezboardman.cgi`

- **`/mail(file|seek)\.cgi/U`**

この例では、URI データ内の `mail` の後に `file` と `seek` のどちらかが続く インスタンスをパケット ペイロードで検索します。

この例は以下のものと一致します。

- `mailfile.cgi`
- `mailseek.cgi`

この例は、以下のものとは**一致しません**。

- `MailFile.cgi`
- `mailfilefile.cgi`

- **m?http\\x3a\x2f\x2f.*(\n|\t)+?U**

この例では、任意の数の文字の後ろにある、HTTP 要求内のタブまたは改行文字を示す URI コンテンツをパケット ペイロード内で検索します。この例では、式で `m?regex?` を使用して、`http\:\/\/` を使用しないようにしています。コロンの前にバックスラッシュがあることに注意してください。

この例は以下のものと一致します。

- `http://www.example.com?scriptvar=x&othervar=\n\..\..`
- `http://www.example.com?scriptvar=\t`

この例は、以下のものとは一致**しません**。

- `ftp://ftp.example.com?scriptvar=&othervar=\n\..\..`
- `http://www.example.com?scriptvar=|/bin/sh -i|`

- **m?http\\x3a\x2f\x2f.*=\|.*\|+?sU**

この例では、（改行を含む）任意の数の文字の後に 1 つの等号、さらに任意の数の文字または空白を含むパイプ文字が続くという構成の URL をパケット ペイロード内で検索します。この例では、式で `m?regex?` を使用して、`http\:\/\/` を使用しないようにしています。

この例は以下のものと一致します。

- `http://www.example.com?value=|/bin/sh/ -i|`
- `http://www.example.com?input=|cat /etc/passwd|`

この例は、以下のものとは**一致しません**。

- `ftp://ftp.example.com?value=|/bin/sh/ -i|`
- `http://www.example.com?value=x&input?|cat /etc/passwd|`

- **/[0-9a-f]{2}\:[0-9a-f]{2}\:[0-9a-f]{2}\:[0-9a-f]{2}\:[0-9a-f]{2}\:[0-9a-f]{2}/i**

この例では、MAC アドレスをパケット ペイロード内で検索します。コロン文字がバックスラッシュでエスケープされていることに注意してください。

metadata キーワード

`metadata` キーワードを使用すると、記述情報をルールに追加できます。また、`metadata` キーワードを `service` 引数とともに使用すると、ネットワーク トラフィック内のアプリケーションとポートを特定することができます。追加する情報を使用して、要件に適合するルールを編成または識別することができ、追加する情報や `service` 引数についてルールを検索することができます。

システムは次の形式の引数に基づいてメタデータを検証します。

key value

ここで、*key* と *value* は、スペースで区切られた記述の組み合わせです。これは、Cisco 提供のルールにメタデータを追加するために Talos インテリジェンスグループ VRT で使用されている形式です。

または、次の形式を使用することもできます。

key = *value*

たとえば、*key value* 形式で次のようにカテゴリとサブカテゴリを使用し、作成者と日付によってルールを識別できます。

```
author SnortGuru_20050406
```

1 つのルール内で複数の `metadata` キーワードを使用できます。また、以下の例に示すように、単一の `metadata` キーワード内で複数の *key value* 引数をカンマで区切ることもできます。

```
author SnortGuru_20050406, revised_by SnortUser1_20050707,
revised_by SnortUser2_20061003,
revised_by SnortUser1_20070123
```

使用できる形式は *key value* と *key*=*value* だけに限定されません。ただし、これらの形式に基づく検証に起因する制限事項を把握しておく必要があります。

注意すべき制限のある文字

次の文字制限に注意してください。

- セミコロン（;）またはコロン（:）を使用しないでください。
- システムはコンマを、複数の *key value* 引数または *key*=*value* 引数の区切り文字であると解釈します。次に例を示します。

  key value,*key value*,*key value*

- システムは等号（=）または余白文字を、*key* と *value* の間の区切り文字であると解釈します。次に例を示します。

  key value

  key=*value*

その他のすべての文字が使用可能です。

注意すべき予約済みメタデータ

`metadata` キーワードでは、次の単語を単一の引数として、または *key value* 引数内の *key* として使用しないでください。これらは Talos 用に予約されています。

```
application
engine
impact_flag
os
policy
rule-type
```

```
rule-flushing
soid
```

（注） ローカル ルールを適切に機能させるために制限付きメタデータをどうしても追加する必要がある場合は、サポート担当にお問い合わせください。

影響レベル 1

`metadata` キーワードでは、次に示す予約済み *key value* 引数を使用できます。

```
impact_flag red
```

この *key value* 引数は、インポートしたローカル ルールまたは侵入ルール エディタを使って作成したカスタム ルールに関する影響フラグを赤（レベル 1）に設定します。

「送信元または宛先のホストがウイルス、トロイの木馬、その他の有害ソフトウェアによって侵害されている可能性があることを、ルールをトリガーしているパケットが示している」と Talos が判断した場合、Talos は Cisco 提供のルールに `impact_flag red` 引数を含めます。

サービス メタデータ

システムは、ネットワークのホストで動作しているアプリケーションを検出し、ネットワーク トラフィックにアプリケーション プロトコル情報を挿入します。これは、検出ポリシーの設定に関係なく実行されます。TCP または UDP ルールで `metadata` キーワード `service` 引数を使用して、ネットワーク トラフィックのアプリケーション プロトコルとポートを照合することができます。ルールで 1 つ以上の `service` アプリケーション引数を単一のポート引数と組み合わせることができます。

サービス アプリケーション

`metadata` キーワードとともに `service` を *key* として、アプリケーションを *value* として使用し、パケットを識別されたアプリケーション プロトコルと一致させることができます。たとえば、次に示す `metadata` キーワード内の *key value* 引数は、ルールを HTTP トラフィックに関連付けます。

```
service http
```

複数のアプリケーションをカンマで区切って指定することもできます。次に例を示します。

```
service http, service smtp, service ftp
```

注意 侵入ルールでサービス メタデータを使用するためには、適応型プロファイルの設定 で説明されているように、アダプティブ プロファイルを有効（デフォルト状態）にする**必要があります**。

次の表に、`service` キーワードとともに使用される最も一般的なアプリケーション値を示します。

（注） 表にないアプリケーションを特定することが難しい場合は、サポートにお問い合わせください。

表 *34 : service* 値

値	説明
cvs	Concurrent Versions System（バージョン管理システム）
dcerpc	分散コンピューティング環境/リモート プロシージャ コール システム
dns	ドメイン ネーム システム
finger	Finger ユーザー情報プロトコル
FTP	ファイル転送プログラム
ftp-data	ファイル転送プログラム（データ チャネル）
http	ハイパーテキスト転送プロトコル
imap	Internet Message Access Protocol
isakmp	Internet Security Association and Key Management Protocol
mysql	My Structured Query Language（構造化照会言語）
netbios-dgm	NETBIOS Datagram Service
netbios-ns	NETBIOS Name Service
netbios-ssn	NETBIOS Session Service
nntp	Network News Transfer Protocol
oracle	Oracle Net Services
shell	OS Shell
pop2	Post Office Protocol バージョン 2
pop3	Post Office Protocol バージョン 3
smtp	Simple Mail Transfer Protocol
snmp	簡易ネットワーク管理プロトコル
ssh	セキュア シェル ネットワーク プロトコル
sunrpc	Sun リモート プロシージャ コール プロトコル

値	説明
telnet	Telnet ネットワーク プロトコル
tftp	トリビアル ファイル転送プロトコル
x11	X Window システム

サービス ポート

Metadata キーワードとともに service を *key* として、指定したポート引数を *value* として使用し、ルールがアプリケーションと組み合わせてポートを照合する方法を定義できます。

次の表の任意のポート値を、ルールごとに 1 つ指定できます。

表 *35 : service* ポート値

値	説明
else-ports または unknown	次の条件のいずれかが満たされるとルールが適用されます。 • パケット アプリケーションが既知で、ルール アプリケーションと一致する。 • パケット アプリケーションが不明で、パケット ポートがルール ポートと一致する。 else-ports および unknown の値では、service がポート修飾子なしでアプリケーションプロトコルを指定する場合にシステムで使用されるデフォルトの動作が生成されます。
and-ports	パケットアプリケーションが既知で、ルールアプリケーションと一致し、パケット ポートがルール ヘッダーのポートと一致する場合、ルールが適用されます。アプリケーションを指定しないルールで and-ports を使用することはできません。
or-ports	次の条件のいずれかが満たされるとルールが適用されます。 • パケット アプリケーションが既知で、ルール アプリケーションと一致する。 • パケット アプリケーションが不明で、パケット ポートがルール ポートと一致する。 • パケット アプリケーションはルール アプリケーションと一致せず、パケット ポートはルール ポートと一致する。 • ルールはアプリケーションを指定せず、パケット ポートはルール ポートと一致する。

次の点に注意してください。

- service アプリケーション引数を service and-ports 引数とともに含める必要があります。
- ルールで上記の表の値が複数指定されている場合、ルールの一番最後にある値が適用されます。
- ポートおよびアプリケーション引数は任意の順序にすることができます。

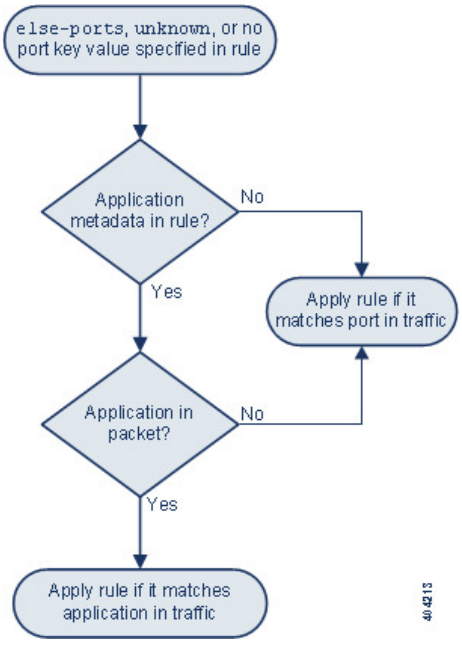
and-ports 値を除き、1 つ以上の service アプリケーション引数の有無にかかわらず、service ポート引数を含めることができます。次に例を示します。

```
service or-ports, service http, service smtp
```

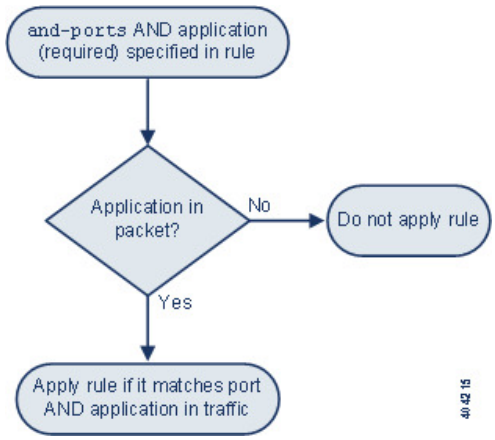
トラフィックのアプリケーションとポート

次の図は、侵入ルールでサポートされるアプリケーションとポートの組み合わせ、およびパケット データにこれらのルール制約を適用した結果を示しています。

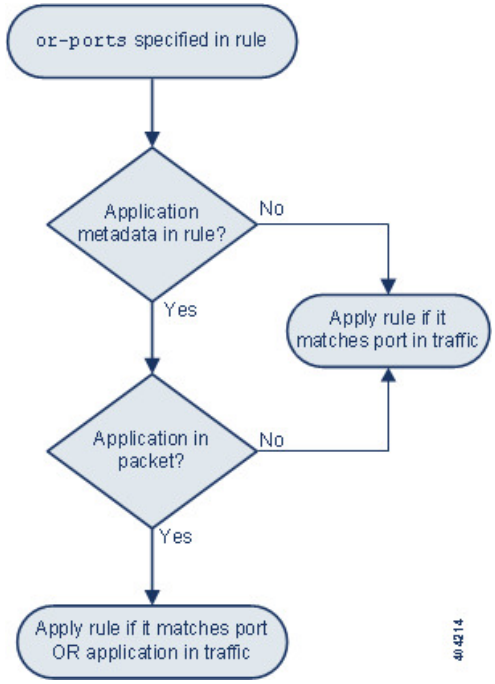
ホスト アプリケーション プロトコル else 送信元/宛先ポート：



ホスト アプリケーション プロトコル **and** 送信元/宛先ポート：



ホスト アプリケーション プロトコル **or** 送信元/宛先ポート：



一致する例

`metadata` キーワードを `service` 引数とともに使用した次のサンプル ルールを、一致するデータおよび一致しないデータの例とともに示します。

- `alert tcp any any -> any [80,8080] (metadata:service and-ports, service http, service smtp;)`

一致する例	一致しない例
• TCP ポート 80 経由の HTTP トラフィック • TCP ポート 8080 経由の HTTP トラフィック • TCP ポート 80 経由の SMTP トラフィック • TCP ポート 8080 経由の SMTP トラフィック	• ポート 80 または 8080 の POP3 トラフィック • ポート 80 または 8080 の不明なアプリケーション トラフィック • ポート 9999 の HTTP トラフィック

- `alert tcp any any -> any [80,8080] (metadata:service or-ports, service http;)`

一致する例	一致しない例
• あらゆるポートの HTTP トラフィック • ポート 80 の SMTP トラフィック • ポート 8080 の SMTP トラフィック • ポート 80 および 8080 の不明なアプリケーションのトラフィック	• 80 または 8080 以外のポートの非 HTTP および非 SMTP トラフィック

- 次のいずれかの規則：
  - `alert tcp any any -> any [80,8080] metadata:service else-ports, service http;)`
  - `alert tcp any any -> any [80,8080] metadata:service unknown, service http;)`
  - `alert tcp any any -> any [80,8080] metadata:service http;)`

一致する例	一致しない例
• あらゆるポートの HTTP トラフィック • パケット アプリケーションが不明な場合はポート 80 • パケット アプリケーションが不明な場合はポート 8080	• ポート 80 または 8080 の SMTP トラフィック • ポート 80 または 8080 の POP3 トラフィック

メタデータ検索のガイドライン

`metadata` キーワードを使用するルールを検索するには、ルールの [検索（Search）] ページで `metadata` キーワードを選択して、オプションで、メタデータの一部分を入力します。たとえば次のように入力できます。

- `search` と入力すると、*key* として `search` が使用されているすべてのルールが表示されます。
- `search http` と入力すると、*key* として `search`、*value* として `http` がそれぞれ使用されているすべてのルールが表示されます。
- `author snortguru` と入力すると、*key* として `author`、*value* として `SnortGuru` がそれぞれ使用されているすべてのルールが表示されます。
- `author s` と入力すると、*key* として `author`、さらに *value* として `SnortGuru`、`SnortUser1`、`SnortUser2` などの語が使用されているすべてのルールが表示されます。

ヒント *key* と *value* の両方を検索するときには、ルール内の *key value* 引数で使用されているのと同じ接続演算子（等号 [=] または空白文字）を検索で使用してください。*key* の後に等号（=）と空白文字のどちらを入力するかに応じて、異なる結果が検索で返されます。

なお、メタデータ追加のために使用する形式とは無関係に、システムはメタデータ検索語を *key value* または *key=value* 引数の全部または一部として解釈します。たとえば、次に示すメタデータは *key value* または *key=value* 形式に従っていませんが、有効なメタデータです。

```
ab cd ef gh
```

ただし、この例に含まれる各スペースは *key* と *value* の間の区切り文字としてシステムで解釈されます。次に示す並列語や単一語を検索で使用すると、この例のメタデータを含むルールを正しく検出できます。

```
cd ef
ef gh
ef
```

一方、次の検索を使用した場合、単一の *key value* 引数としてシステムによって解釈されるため、ルールを検出できません。

```
ab ef
```

関連トピック

ルールの検索 （22 ページ）

IP ヘッダー値

キーワードを使用すると、パケットの IP ヘッダーの中で攻撃やセキュリティ ポリシー違反の可能性を識別できます。

fragbits

fragbits キーワードは、IP 見出し内のフラグメント ビットと予約ビットを検査します。パケットごとに、予約ビット、More Fragments ビット、および Don't Fragment ビットを任意に組み合わせて検査できます。

表 *36 : Fragbits* 引数の値

引数	説明
R	予約済みビット
M	More Fragments ビット
D	Don’t Fragment ビット

fragbits キーワードを使ってルールを微調整するために、次の表に示す演算子をルール内の引数値の後ろに指定できます。

表 *37 : Fragbit* 演算子

演算子	説明
プラス記号（**+**）	パケットは、指定されたすべてのビットと一致する必要があります。
アスタリスク（*****）	パケットは、指定されたどのビットと一致することもできます。
感嘆符（**!**）	指定されたどのビットも設定されていない場合、パケットが基準を満たします。

たとえば、（他のビットの有無とは無関係に）少なくとも予約済みビットが設定されたパケットに対してイベントを生成するには、fragbits 値として R+ を使用します。

id

id キーワードは、キーワード引数で指定される値に照らして IP 見出しフラグメント識別フィールドを検査します。一部のサービス拒否ツールやスキャナは、このフィールドを、容易に検出できる特定の番号に設定します。たとえば、Synscan ポートスキャンを検出する SID 630 では、id 値が 39426（スキャナから伝送されるパケットの ID 番号として使われる静的な値）に設定されます。

（注） id 引数値は数値でなければなりません。

ipopts

IPopts キーワードを使用すると、指定された IP 見出しオプションをパケット内で検索できます。次の表に、使用可能な引数値を示します。

表 *38 : IPoption* 引数

引数	説明
rr	経路を記録
eol	リストの末尾
nop	オペレーションなし
ts	タイム スタンプ
sec	IP セキュリティ オプション
lsrr	厳密でない送信元ルーティング
ssrr	厳密な送信元ルーティング
satid	ストリーム識別子

アナリストが最も頻繁に監視するのは、厳密な送信元ルーティングと厳密でない送信元ルーティングです。これらのオプションは送信元 IP アドレスのスプーフィングを示している可能性があるためです。

ip_proto

`ip_proto` キーワードを使用すると、キーワードの値として指定された IP プロトコルを含むパケットを識別できます。IP プロトコルは 0 ～ 255 の数値として指定できます。これらの番号を、<、>、または ! 演算子と組み合わせることができます。たとえば、ICMP 以外のプロトコルを使用しているトラフィックを検査するには、`ip_proto` キーワードの値として `!1` を使用します。1 つのルール内で `ip_proto` キーワードを複数回にわたって使用できます。ただし、ルール エンジンはキーワードの複数インスタンスをブール和関係（AND）と解釈することに注意してください。たとえば、`ip_proto:!3; ip_proto:!6` を含むルールを作成した場合、このルールは GGP プロトコルおよび TCP プロトコルを使用するトラフィックを無視します。

tos

一部のネットワークでは、ネットワーク上を移動するパケットの優先度を設定するタイプオブサービス（ToS）値が使用されます。`tos` キーワードを使用すると、キーワードの引数で指定された値に照らしてパケットの IP 見出し ToS 値を検査できます。`tos` キーワードを使用するルールは、ToS が指定の値に設定され、しかもルール内の残りの基準を満たすパケットに対してトリガーとして使用されます。

（注） `tos` の引数値は数値でなければなりません。

[ToS] フィールドは IP ヘッダー プロトコルでは非推奨になり、[Differentiated Services Code Point (DSCP)] フィールドに置き換えられています。

ttl

パケットの存続可能時間（time-to-live、ttl）値は、パケットが破棄される前に生成できるホップ数を示します。`ttl` キーワードを使用すると、キーワードの引数として指定された値または値の範囲に照らしてパケットの IP 見出し ttl 値を検査できます。`ttl` キーワード パラメータを 0や1などの低い値に設定すると役立つことがあります。これは、低い存続可能時間値がトレースルートや侵入回避の試みを示している場合があるためです。（ただし、このキーワードの適切な値は、管理対象デバイスの配置やネットワーク トポロジによって異なります）。次のように構文を使用します。

- TTL 値に特定の 1 つの値を設定するには、0 ～ 255 の整数を使用します。値の前に等号（=）を付けることもできます（たとえば `5` または `=5` を指定できます）。
- TTL 値の範囲を指定するには、ハイフン（-）を使用します（たとえば、`0-2` は 0 ～ 2 のすべての値、`-5` は 0 ～ 5 のすべての値、`5-` は 5 ～ 255 のすべての値をそれぞれ指定します）。
- 特定の値より大きい TTL 値を指定するには、「大なり」記号（>）を使用します（たとえば、`>3` は 3 より大きいすべての値を指定します）。
- 特定の値以上の TTL 値を指定するには、「大なりイコール」記号（>=）を使用します（たとえば、`>=3` は 3 以上のすべての値を指定します）。
- 特定の値より小さい TTL 値を指定するには、「小なり」記号（<）を使用します（たとえば、`<3` は 3 より小さいすべての値を指定します）。
- 特定の値以下の TTL 値を指定するには、「小なりイコール」記号（<=）を使用します（たとえば、`<=3` は 3 以下のすべての値を指定します）。

ICMP ヘッダー値

システムでサポートされるキーワードを使用すると、ICMP パケットのヘッダー内の攻撃やセキュリティポリシー違反を識別できます。なお、ほとんどの ICMP タイプおよびコードを検出する事前定義ルールがあることに注意してください。既存のルールを有効にするか、既存のルールに基づいてローカル ルールを作成することを考慮してください。ICMP ルールを最初から作成するよりも、ニーズを満たすルールを見つける方が時間の節約になる可能性があります。

icmp_id と icmp_seq

ICMP の識別番号とシーケンス番号は、ICMP 応答と ICMP 要求を関連付けるうえで役立ちます。通常のトラフィックでは、これらの値はパケットに動的に割り当てられます。一部のコバート チャネルおよび Distributed Denial of Server（DDoS）プログラムは、静的な ICMP ID およびシーケンス値を使用します。次のキーワードを使用すると、静的な値を含む ICMP パケットを識別できます。

キーワード	定義
icmp_id	ICMP エコー要求または応答パケットの ICMP ID 番号を検査します。ICMP ID 番号に対応する数値を `icmp_id` キーワードの引数として使用します。
icmp_seq	`icmp_seq` キーワードは、ICMP エコー要求または応答パケットの ICMP シーケンスを検査します。ICMP シーケンス番号に対応する数値を `icmp_seq` キーワードの引数として使用します。

itype

`itype` キーワードを使用して、特定の ICMP メッセージ タイプ値を含むパケットを検索します。有効な ICMP タイプ値と無効な ICMP タイプ値のいずれかを指定して、さまざまなタイプのトラフィックを検査できます。たとえば、サービス拒否攻撃やフラッディング攻撃を発生させるために攻撃者が範囲外の ICMP タイプ値を設定することがあります。

「小なり」（<）と「大なり」（>）を使用して `itype` 引数値の範囲を指定できます。

次に例を示します。

- <35
- >36
- 3<>55

icode

ICMP メッセージには、宛先が到達不能である場合の詳細を示すコード値が含まれることがあります。

`icode` キーワードを使用すると、特定の ICMP コード値を含むパケットを識別できます。有効な ICMP コード値と無効な ICMP コード値のいずれかを指定することにより、さまざまなタイプのトラフィックを検査できます。

「小なり」（<）と「大なり」（>）を使用して `icode` 引数値の範囲を指定できます。

次に例を示します。

- 35 より小さい値を検索するには `<35` と指定します。
- 36 より大きい値を検索するには `>36` と指定します。
- 3 ～ 55 の間にある値を検索するには、`3<>55` と指定します。

ヒント　`icode` キーワードと `itype` キーワードを一緒に使用すると、両方に一致するトラフィックを識別できます。たとえば、ICMP 宛先到達不能コード タイプと ICMP ポート到達不能コード タイプを含む ICMP トラフィックを特定するには、値 3 の `itype` キーワード（宛先到達不能）と、値 3 の `icode` キーワード（ポート到達不能）を指定します。

TCP ヘッダー値とストリーム サイズ

システムでは、パケットの TCP ヘッダーと TCP ストリームサイズを使って試行される攻撃を識別するためのキーワードを使用できます。

ack

`ack` キーワードを使用すると、パケットの TCP 確認応答番号と特定の値を比較できます。パケットの TCP 確認応答番号が、`ack` キーワードに指定された値と一致した場合に、ルールがトリガーとして使用されます。

`ack` の引数値は数値でなければなりません。

フラグ（Flags）

`flags` キーワードを使用すると、複数の TCP フラグを任意に組み合わせて指定できます。検査対象のパケットでこれらが設定されている場合、ルールがトリガーとして使用されます。

（注） 従来、`flags` の値として `A+` を使用していたケースでは、代わりに `flow` キーワードおよび値 `established` を使用してください。一般に、フラグのすべての組み合わせが検出されるようにするには、フラグの使用時に `flow` キーワードおよび値 `stateless` を使用する必要があります。

次の表に示す `flags` キーワードの値を確認または無視することができます。

表 *39 : flags* の引数

引数	TCP フラグ
ACK	データを確認応答します。
Psh	このパケットでデータが送信される必要があります。
Syn	新しい接続。
Urg	パケットに緊急データが含まれています。
Fin	接続が閉じられました。
Rst	接続が異常終了しました。
CWR	ECN 輻輳ウィンドウが減少しました。旧 R1 引数（下位互換性を維持するために引き続きサポートされています）。
ECE	ECN エコー。旧 R2 引数（下位互換性を維持するために引き続きサポートされています）。

`flags` キーワードを使用する場合、複数のフラグに対する照合方法をシステムに指示するための演算子を使用できます。次の表に、これらの演算子の説明を示します。

表 *40 : flags* と一緒に使用する演算子

演算子	説明	例
すべて	パケットは、指定されたすべてのフラグを含んでいる必要があります。	`Urg` と `all` を選択すると、パケットが緊急フラグを含んでいる必要があること、および他のフラグが含まれる可能性があることを指定できます。
any	パケットは、指定された任意のフラグを含むことができます。	`Ack`、`Psh`、および `any` を選択すると、ルールをトリガーとして使用するためには `Ack` と `Psh` のどちらか（または両方）のフラグが設定される必要があること、およびパケット内で他のフラグも設定されている可能性があることを指定できます。
ノット	パケットは、指定されたフラグセットを**含んではなりません**。	`Urg` と `not` を選択すると、このルールをトリガーとして使用するパケットに関して緊急フラグが設定**されない**ことを指定できます。

flow

`flow` キーワードを使用すると、セッション特性に基づいてルールで検査されるパケットを選択できます。`flow` キーワードを使用することで、ルールの適用対象となるトラフィック フロー方向を指定して、クライアント フローとサーバ フローのどちらかにルールを適用できます。`flow` キーワードによるパケット検査の方法を指定するには、分析すべきトラフィックの方向、検査するパケットの状態、およびパケットが再構築ストリームの一部かどうかを設定できます。

ルールの処理時に、パケットのステートフルインスペクションが実行されます。ステートレス トラフィック（セッション コンテキストが確立されていないトラフィック）を TCP ルールで無視するには、`flow` キーワードをルールに追加して、そのキーワードで **Established** 引数を選択する必要があります。UDP ルールでステートレス トラフィックを無視するには、`flow` キーワードをルールに追加して、**Established** 引数と方向引数のどちらか（または両方）を選択する必要があります。これにより、TCP または UDP ルールでパケットのステートフル インスペクションが実行されます。

方向引数を追加した場合、ルールエンジンは、指定された方向と一致するフローを伴う確立された状態のパケットだけを検査します。たとえば、TCP または UDP 接続が検出されたときトリガーとして使用されるルールに、`flow` キーワードおよび `established` 引数と `From Client` 引数を 追加した場合、ルール エンジンはクライアントから送信されたパケットだけを検査します。

ヒント パフォーマンスを最大にするには、必ず TCP ルールまたは UDP セッション ルールに `flow` キーワードを含めてください。

次の表に、`flow` キーワードで指定できるストリーム関連引数の説明を示します。

表 *41 : flow* の状態関連引数

引数	説明
Established	確立された接続でトリガーとして使用されます。
Stateless	ストリーム プロセッサの状態に関係なくトリガーとして使用されます。

次の表に、`flow` キーワードで指定できる方向オプションの説明を示します。

表 *42 : flow* の方向引数

引数	説明
To Client	サーバ応答でトリガーとして使用されます。
To Server	クライアント応答でトリガーとして使用されます。
From Client	クライアント応答でトリガーとして使用されます。
From Server	サーバ応答でトリガーとして使用されます。

`From Server` と `To Client` の機能が同じであること、および `To Server` と `From Client` の機能も同じであることに注意してください。これらのオプションは、ルールに文脈と読みやすさを加味するために提供されています。たとえば、サーバからクライアントへの攻撃を検出するよう設計されたルールを作成する場合は、`From Server` を使用します。一方、クライアントからサーバへの攻撃を検出するように設計されたルールを作成する場合は、`From Client` を使用します。

次の表に、`flow` キーワードで指定できるストリーム関連引数の説明を示します。

表 *43 : flow* のストリーム関連引数

引数	説明
Ignore Stream Traffic	再構築されたストリーム パケットでトリガーとして使用されません。
Only Stream Traffic	再構築されたストリーム パケットでのみトリガーとして使用されます。

たとえば、`flow` キーワードの値として `To Server, Established, Only Stream Traffic` を使用すると、ストリームプリプロセッサで再構築された、確立済みセッションでクライアントからサーバーに移動するトラフィックを検出できます。

seq

`seq` キーワードを使用すると、静的なシーケンス番号値を指定できます。パケットのシーケンス番号が、指定された引数と一致する場合、そのキーワードを含むルールがトリガーとして使

用されます。このキーワードはあまり使用されませんが、静的シーケンス番号付きの生成済みパケットを使用する攻撃やネットワーク スキャンを識別するうえでこれが役立ちます。

window

`window` キーワードを使用すると、特定の TCP ウィンドウ サイズを指定できます。このキーワードを含むルールは、指定された TCP ウィンドウ サイズのパケットが検出されるたびにトリガーされます。このキーワードはあまり使用されませんが、静的 TCP ウィンドウ サイズ付きの生成済みパケットを使用する攻撃やネットワーク スキャンを識別するうえでこれが役立ちます。

stream_size

次に示す形式で、`stream_size` キーワードとストリーム プリプロセッサを組み合わせて使用すると、TCP ストリームのサイズをバイト単位で特定できます。

```
direction,operator,bytes
```

ここで、bytes はバイト数です。引数内の各オプションをカンマ（,）で区切る必要があります。

次の表は、`stream_size` キーワードで指定できる大文字/小文字を区別しない方向オプションを示しています。

表 *44 : stream_size* キーワードの方向引数

引数	説明
client	指定されたストリーム サイズに一致するクライアントからのストリームでトリガーとして使用されます。
server	指定されたストリーム サイズに一致するサーバからのストリームでトリガーとして使用されます。
both	指定されたストリーム サイズに一致するクライアントからのトラフィックとサーバーからのトラフィックの両方によってトリガーとして使用されます。 たとえば `both, >, 200` という引数は、クライアントからのトラフィックが 200 バイトを超え、しかもサーバからのトラフィックが 200 バイトを超えている場合にトリガーとして使用されます。
either	指定されたストリーム サイズに一致するクライアントまたはサーバーからのトラフィック（どちらか先に出現した方）によってトリガーとして使用されます。 たとえば `both, >, 200` という引数は、クライアントからのトラフィックが 200 バイトを超え、しかもサーバからのトラフィックが 200 バイトを超えている場合にトリガーとして使用されます。

次の表に、`stream_size` キーワードで使用できる演算子の説明を示します。

表 *45 : stream_size* キーワードの引数演算子

演算子	説明
=	等しい
!=	等しくない
>	より大きい
<	より少ない
>=	以上
<=	以下

たとえば、クライアントからサーバーに移動する 5001216 バイト以上の TCP ストリームを検出するには、`stream_size` キーワードの引数として `client, >=, 5001216` を使用できます。

stream_reassembly キーワード

`stream_reassemble` キーワードを使用すると、接続での検査対象トラフィックがルールの条件と一致した場合に、1 つの接続の TCP ストリーム再構築を有効/無効にすることができます。オプションで、このキーワードを 1 つのルール内で複数回使用することができます。

ストリーム再構築を有効または無効にするには、次の構文を使用します。

```
enable|disable, server|client|both, option, option
```

次の表に、`stream_reassemble` キーワードで使用できるオプション引数の説明を示します。

表 *46 : stream_reassemble* のオプション引数

引数	説明
noalert	ルールで他にどの検出オプションが指定されているかに関係なく、イベントを生成しません。
fastpath	一致の検出時に残りの接続トラフィックを無視します。

たとえば、次のルールは、HTTP 応答で 200 OK ステータス コードが検出される接続に対してイベントを生成せずに、TCP クライアント側ストリーム再構築を無効にします。

```
alert tcp any 80 -> any any (flow:to_client, established; content: “200 OK”;
stream_reassemble:disable, client, noalert
```

SSL キーワード

SSL ルール キーワードを使用すると、Secure Sockets Layer（SSL）プリプロセッサを呼び出し、暗号化セッションのパケットから SSL のバージョンとセッション状態に関する情報を抽出できます。

SSL または Transport Layer Security（TLS）を使用する暗号化セッションを確立するためにクライアントとサーバが通信するとき、ハンドシェイク メッセージが交換されます。セッション中に伝送されるデータは暗号化されますが、ハンドシェイク メッセージは暗号化されません。

SSL プリプロセッサは、特定のハンドシェイク フィールドから状態とバージョンの情報を抽出します。ハンドシェイク内の 2 つのフィールドは、セッション暗号化に使われる SSL または TLS のバージョンとハンドシェイクのステージを示します。

ssl_state

`ssl_state` キーワードを使用すると、暗号化されたセッションの状態情報と照合することができます。同時に使用される複数の SSL バージョンを検査するには、1 つのルール内で複数の `ssl_version` キーワードを使用します。

ルールで `ssl_state` キーワードが使用されている場合、ルール エンジンは SSL プリプロセッサを呼び出して、トラフィック内の SSL 状態情報を検査します。

たとえば、チャレンジ長が非常に長く、データが多すぎる `ClientHello` メッセージを送信することによってサーバー上のバッファ オーバーフローを引き起そうとする攻撃者の試みを検出するには、`ssl_state` キーワードと引数 `client_hello` を使用し、異常に大きなパケットを検査することができます。

SSL 状態に関する複数の引数を指定するには、カンマ区切りのリストを使用します。複数の引数を列挙した場合、システムは OR 演算子を使ってそれらを評価します。たとえば、引数として `client_hello` および `server_hello` を指定すると、システムは `client_hello` または `server_hello` のどちらかを含むトラフィックに照らしてルールを評価します。

次のように、引数を除外することもできます。

```
!client_hello, !unknown
```

接続が一連の状態のそれぞれに到達したことを確認するには、ssl_state ルール オプションを使用する複数のルールを使う必要があります。`ssl_state` キーワードは、次の識別子を引数として受け入れます。

表 *47: ssl_state* の引数

引数	目的
`client_hello`	クライアントが暗号化セッションを要求する、メッセージ タイプ `ClientHello` のハンドシェイク メッセージを照合します。
`server_hello`	クライアントからの暗号化セッション要求に対してサーバーが応答する、メッセージ タイプ `ServerHello` のハンドシェイク メッセージを照合します。

引数	目的
client_keyx	サーバーからのキーの受信を確認するためにクライアントがサーバーにキーを伝送する、メッセージ タイプ ClientKeyExchange のハンドシェイク メッセージを照合します。
server_keyx	サーバーからのキーの受信を確認するためにクライアントがサーバーにキーを伝送する、メッセージ タイプ ServerKeyExchange のハンドシェイク メッセージを照合します。
unknown	任意のハンドシェイク メッセージ タイプを照合します。

ssl_version

ssl_version キーワードを使用すると、暗号化されたセッションのバージョン情報と照合することができます。ルールで ssl_version キーワードが使用されている場合、ルール エンジンは SSL プリプロセッサを呼び出して、トラフィック内の SSL バージョン情報を検査します。

たとえば、SSL バージョン 2 にバッファ オーバーフロー脆弱性があることがわかっている場合、ssl_version キーワードで sslv2 引数を使用して、その SSL バージョンを使用するトラフィックを識別できます。

SSL バージョンに関する複数の引数を指定するには、カンマ区切りのリストを使用します。複数の引数を列挙した場合、システムは OR 演算子を使ってそれらを評価します。たとえば、SSLv2 を使用していない暗号化トラフィックを識別するには、ssl_version:ssl_v3,tls1.0,tls1.1,tls1.2 をルールに追加できます。このルールは、SSL バージョン 3、TLS バージョン 1.0、TLS バージョン 1.1、または TLS バージョン 1.2 を使用するトラフィックを評価します。

ssl_version キーワードは、次の SSL/TLS バージョン識別子を引数として受け入れます。

表 *48: ssl_version* の引数

引数	目的
sslv2	Secure Sockets Layer（SSL）バージョン 2 を使用してエンコードされたトラフィックを照合します。
sslv3	Secure Sockets Layer（SSL）バージョン 3 を使用してエンコードされたトラフィックを照合します。
tls1.0	Transport Layer Security（TLS）バージョン 1.0 を使用してエンコードされたトラフィックを照合します。
tls1.1	Transport Layer Security（TLS）バージョン 1.1 を使用してエンコードされたトラフィックを照合します。
tls1.2	Transport Layer Security（TLS）バージョン 1.2 を使用してエンコードされたトラフィックを照合します。

appid キーワード

パケットからアプリケーションプロトコル、クライアントアプリケーション、Webアプリケーションを特定するために appid キーワードを使用できます。たとえば、ある脆弱性をもつことが知られている特定のアプリケーションを検出することを考えます。

侵入ルールの appid キーワードの中で、[AppID の設定（Configure AppID）] をクリックし、検出するアプリケーションを 1 つまたは複数選択します。

使用可能なアプリケーションの参照

条件の作成を初めて開始するときは、[使用可能なアプリケーション（Available Applications）] リストは制約されておらず、システムが検出するすべてのアプリケーションをページごとに 100 個ずつ表示します。

- アプリケーションを確認していくには、リストの下にある矢印をクリックします。
- アプリケーションの特性に関する概要情報と参照可能なインターネット検索リンクを含むポップアップウィンドウを表示するには、アプリケーションの横にある[情報（Information）]（ ）をクリックします。

アプリケーション フィルタの使用

照合するアプリケーションを見つけやすくするために、[使用可能なアプリケーション（Available Applications）] リストを次のように制約できます。

- アプリケーションを検索するには、リスト上部にある [名前で検索（Search by name）] プロンプトをクリックし、名前を入力します。入力すると、リストが更新されて一致するアプリケーションが表示されます。
- フィルタを適用してアプリケーションを制約するには、[アプリケーション フィルタ（Application Filters）] リストを使用します。フィルタを適用すると、[使用可能なアプリケーション（Available Applications）] リストが更新されます。便宜上、システムは**ロック解除アイコン**を使用して、復号されたトラフィック（暗号化されているトラフィックまたは暗号化されていないトラフィックではなく）でのみ識別できるアプリケーションをマークします。

（注） [アプリケーション フィルタ（Application Filters）] リストで 1 つ以上のフィルタを選択し、しかも [使用可能なアプリケーション（Available Applications）] リストを検索した場合、選択内容と検索フィルタ適用後の [使用可能なアプリケーション（Available Applications）] リストが AND 演算を使って結合されます。

アプリケーションの選択

アプリケーションを 1 つだけ選択するには、そのアプリケーションを選択し、[ルールへの追加（Add to Rule）] をクリックします。フィルタで限定されている現在の表示のすべてのアプリケーションを選択するには、右クリックして [すべて選択（Select All）] を選択します。

アプリケーション層プロトコル値

アプリケーション層プロトコル値の正規化と検査はほとんどがプリプロセッサによって実行されますが、種々のプリプロセッサ オプションを使用して、アプリケーション層値をさらに検査できます。

RPC キーワード

`rpc` キーワードは、TCP または UDP パケットでオープン ネットワーク コンピューティング リモート プロシージャ コール（ONC RPC）サービスを識別します。これにより、ホスト上の RPC プログラムの識別試行を検出することができます。ネットワークで実行中のいずれかの RPC サービスを悪用できるかどうか判断するために、侵入者は RPC ポートマッパーを使用できます。また、ポートマッパーを使用せずに RPC を実行中の他のポートへのアクセスを試みることもできます。次の表に、`rpc` キーワードで使用できる引数を列挙します。

表 *49 : rpc* キーワードの引数

引数	説明
アプリケーション	RPC アプリケーション番号
手順	呼び出される RPC プロシージャ
version	RPC バージョン

`rpc` キーワードの引数を指定するには、次の構文を使用します。

```
application,procedure,version
```

ここで、application は RPC アプリケーション番号、procedure は RPC プロシージャ番号、version は RPC バージョン番号です。`rpc` キーワードのすべての引数を指定する必要があります。引数のいずれかを指定できない場合は、アスタリスク（`*`）で置き換えてください。

たとえば、任意のプロシージャまたはバージョンの RPC ポートマッパー（100000 という番号で示される RPC アプリケーション）を検索するには、引数として `100000,*,*` を使用します。

ASN.1 キーワード

`asn1` キーワードを使用すると、さまざまな有害エンコードを検索しながら、パケットまたはパケットの一部分をデコードできます。

次の表に、`asn1` キーワードの引数について説明します。

表 *50 : asn.1* キーワードの引数

引数	説明
Bitstring Overflow	無効な、リモートで悪用可能なビットストリング エンコードを検出します。
Double Overflow	標準バッファより大きい二重 ASCII エンコードを検出します。これは Microsoft Windows の悪用可能な機能であることが知られていますが、現時点でどのサービスが悪用可能であるかは不明です。
Oversize Length	指定された引数より大きい ASN.1 タイプ長を検出します。たとえば Oversize Length を 500 に設定した場合、500 を上回る ASN.1 タイプによってルールがトリガーとして使用されます。
Absolute Offset	パケット ペイロードの先頭からの絶対オフセットを設定します（offset カウンタがバイト 0 から始まることに注意してください）。たとえば SNMP パケットをデコードするには、Absolute Offset を 0 に設定し、Relative Offset を設定しません。Absolute Offset として正または負の値が可能です。
Relative Offset	これは、最後に見つかったコンテンツ一致、`pcre`、または `byte_jump` からの相対オフセットです。コンテンツ "foo" の直後の ASN.1 シーケンスをデコードするには、Relative Offset を 0 に設定し、Absolute Offset を設定しません。Relative Offset として正または負の値が可能です。（オフセット カウンタが 0 から始まることに注意してください。）

たとえば、Microsoft ASN.1 ライブラリにおける既知の脆弱性ではバッファ オーバーフローが発生し、攻撃者は特別に細工した認証パケットを使ってその状態を悪用できます。システムが asn.1 データをデコードするとき、パケット内の exploit コードは、システム レベル特権付きでホスト上で動作したり、DoS 状態を引き起したりすることができます。次のルールは、`asn1` キーワードを使用して、この脆弱性を悪用する試みを検出します。

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 445
(flow:to_server, established; content:"|FF|SMB|73|";
nocase; offset:4; depth:5;
asn1:bitstring_overflow,double_overflow,oversize_length 100,
relative_offset 54;)
```

上記のルールの場合、任意のポートおよび $EXTERNAL_NET 変数で定義された任意の IP アドレスから発信され、ポート 445 を使用する $HOME_NET 変数で定義された任意の IP アドレスに向かう TCP トラフィックに対して、イベントが生成されます。加えて、サーバへの TCP 接続が確立された時点でのみルールを実行します。その後、ルールは特定の位置にある特定のコンテンツを検査します。最後に、ルールは `asn1` キーワードを使用して、ビットストリング エンコードと二重 ASCII エンコードを検出し、最後に見つかったコンテンツ一致の末尾から 55 バイト目以降、長さ 100 バイトを超える asn.1 タイプ長を識別します（`offset` カウンタがバイト 0 から始まることに注意してください。）

urilen キーワード

urilen キーワードと HTTP Inspect プリプロセッサを組み合わせて使用すると、特定の長さ、最大長を下回る、最小長を上回る、または指定された範囲内の URI を HTTP トラフィック内で検査できます。

HTTP Inspect プリプロセッサがパケットを正規化して検査した後、ルール エンジンはルールに照らしてそのパケットを評価し、urilen キーワードで指定された長さ条件に URI が一致するかどうか判断します。このキーワードを使用すると、URI 長の脆弱性をエクスプロイトしようとする試みを検出できます。たとえばバッファ オーバーフローを発生させて、攻撃者が DoS 状態を引き起こしたり、システム レベル特権付きでホスト上でコードを実行したりしようと試みる可能性があります。

ルール内で urilen キーワードを使用するときには、次の点に注意してください。

- 必ず flow:established キーワードおよび他の 1 つ以上のキーワードを組み合わせて、urilen キーワードを使用してください。
- ルール プロトコルは常に TCP です。
- ターゲット ポートは常に HTTP ポートです。

URI 長を指定するときには、10 進のバイト数、「小なり」（<）、および「大なり」（>）を使用します。

次に例を示します。

- 5 バイト長の URI を検出するには、5 を指定します。
- 5 バイト長を下回る URI を検出するには、< 5（1 つの空白文字で区切る）を指定します。
- 5 バイト長を上回る URI を検出するには、> 5（1 つの空白文字で区切る）を指定します。
- 3 ～ 5 バイト長の URI を検出するには、3 <> 5（<> の前後に空白文字を 1 つずつ含む）を指定します。

たとえば、Novell の eDirectory バージョン 8.8 に付属のサーバー モニタリングおよび診断ユーティリティ iMonitor バージョン 2.4 に脆弱性があることが知られています。長すぎる URI を含むパケットはバッファ オーバーフローを発生させるため、攻撃者はシステム レベル特権付きでホスト上で動作したり、DoS 状態を引き起こしたりできる特別に細工したパケットを使ってその状態をエクスプロイトできます。次のルールは、urilen キーワードを使用して、この脆弱性を悪用する試みを検出します。

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT eDirectory 8.8 Long URI iMonitor buffer
overflow attempt"; flow:to_server,established;
urilen:> 8192; uricontent:"/nds/"; nocase;
classtype:attempted-admin; sid:x; rev:1;)
```

上記のルールの場合、任意のポートおよび $EXTERNAL_NET 変数で定義された任意の IP アドレスから発信され、$HTTP_PORTS 変数で定義されたポートを使用して、$HOME_NET 変数で

定義された任意のIPアドレスに向かうTCPトラフィックに対して、イベントが生成されます。加えて、サーバーへの TCP 接続が確立された時点でのみ、パケットがルールに照らして評価されます。ルールは、`urilen` キーワードを使用して、長さ 8192 バイトを超える URI を検出します。最後に、ルールはURIを検索して、大文字/小文字を区別しない特定のコンテンツ `/nds/` を探します。

関連トピック

侵入ルール ヘッダー プロトコル （5 ページ）
侵入ルール ヘッダーの送信元および宛先ポート （9 ページ）
定義済みデフォルト変数

DCE/RPC キーワード

次の表で説明する 3 つの DCE/RPC キーワードを使用して、DCE/RPC セッション トラフィックのエクスプロイトをモニタできます。これらのキーワードを含むルールを処理するとき、システムは DCE/RPC プリプロセッサを呼び出します。

表 *51 : DCE/RPC* キーワード

使用するフィルタ	使用方法	検出対象
`dce_iface`	単独	特定のDCE/RPCサービスを特定するパケット
`dce_opnum`	`dce_iface` の後ろ	特定の DCE/RPC サービス オペレーションを特定するパケット
`dce_stub_data`	`dce_iface` + `dce_opnum` の後ろ	特定の処理要求または応答を定義するスタブ データ

表に示されているように、`dce_opnum` の前に必ず `dce_iface` を配置し、`dce_stub_data` の前に必ず `dce_iface` + `dce_opnum` を配置する必要があることに注意してください。

また、これらの DCE/RPC キーワードを他のルール キーワードと組み合わせて使用することもできます。DCE/RPC ルールでは、**DCE/RPC** の引数が選択された状態で `byte_jump`、`byte_test`、`byte_extract` の各キーワードを使用することに注意してください。

シスコでは、DCE/RPC キーワードを含むルールに 1 つ以上の `content` キーワードを含めることを推奨しています。こうすると、ルール エンジンが常に高速パターン マッチ機能を使用することで処理速度が上がり、パフォーマンスが向上します。ルールに 1 つ以上の `content` キーワードが含まれている場合は、`content` キーワードの [高速パターン マッチ機能を使用（Use Fast Pattern Matcher）] 引数が有効になっているかどうかに関係なく、ルール エンジンが高速パターン マッチ機能を使用することに注意してください。

次のケースでは、DCE/RPC バージョンおよび隣接ヘッダー情報を一致コンテンツとして使用できます。

- ルールに他の `content` キーワードが含まれていない
- ルールにもう 1 つ `content` キーワードが含まれているが、DCE/RPC バージョンおよび隣接情報が、他方の content よりも特有のパターンを表している

たとえば、DCE/RPCバージョンおよび隣接情報は通常、1バイトのコンテンツよりも特有です。

次に示すバージョンおよび隣接情報コンテンツ一致のいずれか1つを使用して、ルール限定を終了する必要があります。

- コネクション型DCE/RPCルールでは、コンテンツ `|05 00 00|`（メジャー バージョン05、マイナー バージョン00、および要求PDU（プロトコル データ ユニット）タイプ00）を使用します。
- コネクションレス型DCE/RPCルールでは、コンテンツ `|04 00|`（バージョン04、要求PDUタイプ00）を使用します。

いずれの場合も、DCE/RPC プリプロセッサで完了済みの処理を繰り返すことなく高速パターンマッチ機能を呼び出すために、ルール内の最後のキーワードとしてバージョンおよび隣接情報の `content` キーワードを配置してください。ルールの末尾に配置される `content` キーワードは、高速パターン マッチ機能を呼び出す手段として使われるバージョン コンテンツに当てはまりますが、ルール内の他のコンテンツ一致には必ずしも当てはまらないことに注意してください。

関連トピック

DCE/RPC プリプロセッサ
content キーワードと protected_content キーワード （28 ページ）
content キーワードの高速パターン マッチ機能の引数 （39 ページ）
概要：byte_jump および byte_test キーワード
byte_extract キーワード （48 ページ）

dce_iface

`dce_iface` キーワードを使用すると、特定の DCE/RPC サービスを識別できます。

オプションで、`dce_iface` キーワードを `dce_opnum` キーワードおよび `dce_stub_data` キーワードと組み合わせて使用すると、検査する DCE/RPC トラフィックをさらに限定することができます。

固定型16バイト Universally Unique Identifier（UUID）は、それぞれの DCE/RPC サービスに割り当てられるアプリケーション インターフェイスを識別します。たとえば、UUID 4b324fc8-670-01d3-1278-5a47bf6ee188 は、srvsvc サービスとしても知られる DCE/RPC lanmanserver サービスを識別します。このサービスは、ピアツーピア プリンタ、ファイル、および SMB 名前付きパイプを共有するためのさまざまな管理機能を提供します。DCE/RPC プリプロセッサは UUID および関連する見出し値を使用して DCE/RPC セッションを追跡します。

インターフェイス UUID は、次のように、ハイフンで区切られた 5 つの 16 進文字列で構成されます。

```
<4hexbytes>-<2hexbytes>-<2hexbytes>-<2hexbytes>-<6hexbytes>
```

次に示す netlogon インターフェイスの UUID のように、ハイフンを含む UUID 全体を入力することで、インターフェイスを指定します。

```
12345678-1234-abcd-ef00-01234567cffb
```

UUID 内の最初の 3 つの文字列はビッグ エンディアン バイト順で指定される必要があることに注意してください。通常、公開されたインターフェイス リストやプロトコル アナライザには UUID が正しいバイト順で表示されますが、それを入力する前に UUID バイト順を変更しなければならない場合もあります。次に示すメッセンジャー サービス UUID の場合、リトル エンディアン バイト順の最初の 3 つの文字列を含む未加工 ASCII テキストで表示されることがあります。

```
f8 91 7b 5a 00 ff d0 11 a9 b2 00 c0 4f b6 e6 fc
```

この同じ UUID を `dce_iface` キーワードに指定するには、次のようにハイフンを挿入し、最初の 3 つの文字列をビッグ エンディアン バイト順で配置できます。

```
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc
```

1 つの DCE/RPC セッションに複数のインターフェイスへの要求を含めることができますが、1 つのルールには 1 つの `dce_iface` キーワードだけを含めてください。追加のインターフェイスを検出するには、追加のルールを作成します。

DCE/RPC アプリケーション インターフェイスにはインターフェイス バージョン番号も割り当てられます。オプションで、インターフェイス バージョンを指定できます。その際、バージョンが指定値に等しい、等しくない、指定値より小さい、または大きいことを示す演算子を使用します。

TCP セグメンテーションや IP フラグメンテーションに加えて、コネクション型とコネクションレス型の両方の DCE/RPC をフラグメント化することができます。通常、先頭以外の DCE/RPC フラグメントを指定のインターフェイスに関連付けるのはあまり効率的ではありません。このようにすると、多数の誤検出が発生する可能性があります。ただし、柔軟性を維持するために、オプションで、指定されたインターフェイスに照らしてすべてのフラグメントを評価できます。

次の表に、`dce_iface` キーワードの引数を要約します。

表 *52: dce_iface* の引数

引数	説明
Interface UUID	DCE/RPC トラフィック内で検出対象となる特定のサービスのアプリケーション インターフェイスを識別する、ハイフンを含む UUID。指定されたインターフェイスに関連付けられた任意の要求がインターフェイス UUID に一致します。
バージョン	オプションで、アプリケーション インターフェイス バージョン番号 0 ～ 65535 と、検出対象のバージョンが指定値より大きい（>）、小さい（<）、等しい（=）、または等しくない（!）を示す演算子。

引数	説明
All Fragments	オプションで、関連するすべてのDCE/RPCフラグメント内のインターフェイスの照合、およびインターフェイスバージョン（指定されている場合）での照合を有効にします。この引数はデフォルトで無効になっています。これは、最初のフラグメントまたはフラグメント化されていないパケット全体が指定のインターフェイスに関連付けられている場合にのみ、キーワードが一致することを意味します。この引数を有効にすると、誤検出が発生する可能性があることに注意してください。

dce_opnum キーワード

dce_opnum キーワードを DCE/RPC プリプロセッサと組み合わせて使用すると、DCE/RPC サービスが提供する 1 つ以上の特定のオペレーションを識別するパケットを検出できます。

クライアント関数呼び出しは、DCE/RPC 仕様で「オペレーション」と呼ばれる特定のサービス関数を要求します。オペレーション番号（opnum）はDCE/RPCヘッダー内の特定のオペレーションを識別します。エクスプロイトは特定のオペレーションを標的にすることがあります。

たとえば UUID 12345678-1234-abcd-ef00-01234567cffb は、数十種類のオペレーションを提供する netlogon サービスのインターフェイスを識別します。その 1 つがオペレーション 6（NetrServerPasswordSet オペレーション）です。

オペレーション用のサービスを識別するには、dce_opnum キーワードの前に dce_iface キーワードを指定する必要があります。

特定のオペレーションを示す1つの10進数値（0 ～ 65535 の範囲）、ハイフンで区切られたオペレーション範囲、またはカンマ区切りのオペレーション/範囲リストを任意の順序で指定できます。

次の例は、すべて有効な netlogon オペレーション番号を表しています。

```
15
15-18
15, 18-20
15, 20-22, 17
15, 18-20, 22, 24-26
```

dce_stub_data キーワード

dce_stub_data キーワードをDCE/RPCプリプロセッサと組み合わせて使用すると、他のルールオプションとは無関係に、スタブ データの先頭からインスペクションを開始するようルールエンジンに指示できます。dce_stub_data キーワードの後に続くパケット ペイロード ルール オプションは、スタブ データ バッファを基準にして適用されます。

DCE/RPC スタブ データは、クライアント プロシージャ コールと DCE/RPC ランタイム システム（DCE/RPC の中核をなすルーチンとサービスを提供するメカニズム）の間のインターフェイスを提供します。DCE/RPCエクスプロイトは、DCE/RPCパケットのスタブデータ部分で識別されます。スタブデータは特定のオペレーションまたは関数呼び出しに関連付けられているため、必ず dce_stub_data の前に dce_iface と dce_opnum を指定して、関連するサービスとオペレーションを識別してください。

dce_stub_data キーワードには引数がありません。

SIP キーワード

4 つの SIP キーワードを使用すると、SIP セッション トラフィックでエクスプロイトを監視できます。

SIP プロトコルはサービス拒否（DoS）攻撃に対して脆弱であることに注意してください。このような攻撃に対処するルールでは、レート ベースの攻撃防御を活用できます。

sip_header キーワード

sip_header キーワードを使用すると、抽出された SIP 要求または応答ヘッダーの先頭から検査を開始し、検査対象をヘッダー フィールドに限定することができます。

sip_header キーワードには引数がありません。

次の例のルール フラグメントは SIP ヘッダーを指し示し、CSeq ヘッダー フィールドに一致します。

```
alert udp any any -> any 5060 ( sip_header; content:"CSeq"; )
```

関連トピック

動的侵入ルール状態
レート ベースの攻撃防御

sip_body キーワード

sip_body キーワードを使用すると、抽出された SIP 要求または応答メッセージ本文の先頭から検査を開始し、検査対象をメッセージ本文に限定することができます。

sip_body キーワードには引数がありません。

次の例のルール フラグメントは SIP メッセージ本文を指し示し、抽出された SDP データの c（接続情報）フィールド内の特定の IP アドレスに一致します。

```
alert udp any any -> any 5060 ( sip_body; content:"c=IN 192.168.12.14"; )
```

ルールが SDP コンテンツの検索だけに限定されないことに注意してください。SIP プリプロセッサはメッセージ本文全体を抽出し、それをルール エンジンで使用できるようにします。

sip_method キーワード

各 SIP 要求内の *method* フィールドは要求の目的を識別します。sip_method キーワードを使用すると、SIP 要求の中で特定のメソッドを検査することができます。複数のメソッドはカンマで区切ります。

次に示す現在定義されている SIP メソッドを指定できます。

```
ack, benotify, bye, cancel, do, info, invite, join, message, notify, options, prack,
publish, quath, refer, register, service, sprack, subscribe, unsubscribe, update
```

メソッドでは大文字と小文字が区別されません。複数のメソッドをカンマで区切ることができます。

今後、新しいSIPメソッドが定義される可能性があるため、カスタムメソッド、つまり現在定義されているSIPメソッド以外のメソッドを指定することもできます。可能なフィールド値はRFC 2616で定義されています。=、(、)などの制御文字と区切り文字を除いて、すべての文字を使用できます。除外されている区切り文字の完全なリストについては、RFC 2616を参照してください。指定されたカスタムメソッドがトラフィックで検出されると、システムはパケットヘッダーを検査しますが、メッセージは検査されません。

システムでは最大32個のメソッド（現在定義されている21個のメソッドと追加の11個のメソッド）がサポートされます。システムは、設定される未定義のメソッドをすべて無視します。合計32個のメソッドには、SIPプリプロセッサのオプション[検査するメソッド（Methods to Check）]を使って指定されるメソッドが含まれることに注意してください。

否定を使用する場合は、1つのメソッドだけを指定できます。次に例を示します。

```
!invite
```

ただし、1つのルール内の複数の `sip_method` キーワードが **AND** 演算で結合されることに注意してください。たとえば、`invite` と `cancel` を除くすべての抽出されたメソッドを検査するには、次のような2つの除外付き `sip_method` キーワードを使用します。

```
sip_method: !invite
sip_method: !cancel
```

Ciscoでは、`sip_method` キーワードを含むルールに1つ以上の `content` キーワードを含めることを推奨しています。こうすると、ルールエンジンが常に高速パターンマッチ機能を使用することで処理速度が上がり、パフォーマンスが向上します。ルールに1つ以上の `content` キーワードが含まれている場合は、`content` キーワードの[高速パターンマッチ機能を使用（Use Fast Pattern Matcher）]引数が有効になっているかどうかに関係なく、ルールエンジンが高速パターンマッチ機能を使用することに注意してください。

関連トピック

SIPプリプロセッサのオプション

content キーワードと protected_content キーワード （28 ページ）
content キーワードの高速パターン マッチ機能の引数 （39 ページ）

sip_stat_code キーワード

各SIP応答内の3桁のステータスコードは、要求されたアクションの結果を示します。`sip_stat_code` キーワードを使用すると、SIP応答の中で特定のステータスコードを検査することができます。

1桁の応答タイプ番号1～9、特定の3桁の番号100～999、またはこれらを任意に組み合わせたカンマ区切りリストを指定できます。リスト内のいずれか1つの番号がSIP応答内のコードに一致する場合、そのリストが一致します。

次の表に、指定可能なSIPステータスコード値の説明を示します。

表 *53*：*sip_stat_code* の値

検出対象	指定する内容	例	検出結果
1 つの特定のステータス コード	3 桁のステータス コード	189	189
指定された 1 桁で始まる 3 桁のコード	1 桁	1	1xx、つまり 100、101、102 など
値のリスト	特定のコードおよび 1 桁を任意に組み合わせてカンマで区切る	222, 3	222 および 300、301、302 など

また、ルールに content キーワードが含まれているかどうかに関係なく、sip_stat_code キーワードを使って指定された値を検索するためにルール エンジンが高速パターン マッチ機能を使用しないことにも注意してください。

GTP キーワード

3 つの GSRP トンネリング プロトコル（GTP）キーワードを使用すると、GTP バージョン、メッセージタイプ、および情報要素をコマンドチャネル内で検査できます。contentやbyte_jumpなどの他の侵入ルール キーワードと組み合わせて GTP キーワードを使用することはできません。gtp_info または gtp_type キーワードを使用するそれぞれのルールで、gtp_version キーワードを使用する**必要があります**。

gtp_version キーワード

gtp_version キーワードを使用すると、GTP 制御メッセージの中で GTP バージョン 0、1、または 2 を検査することができます。

定義されているメッセージ タイプと情報要素はGTPバージョンによって異なるため、gtp_type または gtp_info キーワードを使用するときには、gtp_version を使用する必要があります。値として 0、1、または 2 を指定できます。

gtp_type キーワード

それぞれの GTP メッセージは、数値と文字列で構成されるメッセージ タイプによって識別されます。gtp_type キーワードを使用すると、特定の GTP メッセージ タイプのトラフィックを検査できます。定義されているメッセージ タイプと情報要素は GTP バージョンによって異なるため、gtp_type または gtp_info キーワードを使用するときには、gtp_version も使用する必要があります。

次の例に示すように、メッセージ タイプとして定義済みの 10 進数値、定義済み文字列、あるいはどちらか（または両方）を任意に組み合わたカンマ区切りリストを指定できます。

```
10, 11, echo_request
```

リスト内のそれぞれの値または文字列を照合するとき、システムはOR演算を使用します。値と文字列を列挙する順序は重要ではありません。リスト内のいずれか1つの値または文字列の一致により、キーワードが一致します。認識されない文字列または範囲外の値を含むルールを保存しようとすると、エラーが発生します。

表に示されているように、GTP バージョンに応じて、同じメッセージ タイプの値が異なる場合があることに注意してください。たとえば `sgsn_context_request` メッセージ タイプの値は GTPv0 と GTPv1 では 50 ですが、GTPv2 では 130 です。

パケット内のバージョン番号に応じて、`gtp_type` キーワードは異なる値と一致します。上記の例の場合、GTPv0 または GTPv1 パケットではキーワードがメッセージ タイプ値 50 と一致しますが、GTPv2 パケットでは値 130 と一致します。パケット内のメッセージ タイプ値が、パケットで指定されたバージョンの既知の値でない場合は、キーワードがパケットと一致しません。

メッセージタイプに整数を指定した場合、パケット内で指定されたバージョンとは無関係に、キーワード内のメッセージ タイプが GTP パケット内の値と一致すればキーワードが一致します。

次の表に、GTP メッセージ タイプごとにシステムで認識される定義済みの値と文字列を示します。

表 *54 : GTP* メッセージ タイプ

値	バージョン 0	バージョン 1	バージョン 2
1	echo_request	echo_request	echo_request
2	echo_response	echo_response	echo_response
3	version_not_supported	version_not_supported	version_not_supported
4	node_alive_request	node_alive_request	該当なし
5	node_alive_response	node_alive_response	該当なし
[6]	redirection_request	redirection_request	該当なし
7	redirection_response	redirection_response	該当なし
16	create_pdp_context_request	create_pdp_context_request	該当なし
17	create_pdp_context_response	create_pdp_context_response	該当なし
18	update_pdp_context_request	update_pdp_context_request	該当なし
19	update_pdp_context_response	update_pdp_context_response	該当なし
20	delete_pdp_context_request	delete_pdp_context_request	該当なし
21	delete_pdp_context_response	delete_pdp_context_response	該当なし

値	バージョン 0	バージョン 1	バージョン 2
22	create_aa_pdp_context_request	init_pdp_context_activation_request	該当なし
23	create_aa_pdp_context_response	init_pdp_context_activation_response	該当なし
24	delete_aa_pdp_context_request	該当なし	該当なし
25	delete_aa_pdp_context_response	該当なし	該当なし
26	error_indication	error_indication	該当なし
27	pdu_notification_request	pdu_notification_request	該当なし
36	pdu_notification_response	pdu_notification_response	該当なし
29	pdu_notification_reject_request	pdu_notification_reject_request	該当なし
30	pdu_notification_reject_response	pdu_notification_reject_response	該当なし
31	該当なし	supported_ext_header_notification	該当なし
32	send_routing_info_request	send_routing_info_request	create_session_request
33	send_routing_info_response	send_routing_info_response	create_session_response
34	failure_report_request	failure_report_request	modify_bearer_request
35	failure_report_response	failure_report_response	modify_bearer_response
36	note_ms_present_request	note_ms_present_request	delete_session_request
37	note_ms_present_response	note_ms_present_response	delete_session_response
38	該当なし	該当なし	change_notification_request
39	該当なし	該当なし	change_notification_response
48	identification_request	identification_request	該当なし
49	identification_response	identification_response	該当なし
50	sgsn_context_request	sgsn_context_request	該当なし
51	sgsn_context_response	sgsn_context_response	該当なし
52	sgsn_context_ack	sgsn_context_ack	該当なし
53	該当なし	forward_relocation_request	該当なし
54	該当なし	forward_relocation_response	該当なし
55	該当なし	forward_relocation_complete	該当なし

値	バージョン 0	バージョン 1	バージョン 2
72	該当なし	relocation_cancel_request	該当なし
57	該当なし	relocation_cancel_response	該当なし
58	該当なし	forward_srns_contex	該当なし
59	該当なし	forward_relocation_complete_ack	該当なし
60	該当なし	forward_srns_contex_ack	該当なし
64	該当なし	該当なし	modify_bearer_command
65	該当なし	該当なし	modify_bearer_failure_indication
66	該当なし	該当なし	delete_bearer_command
67	該当なし	該当なし	delete_bearer_failure_indication
68	該当なし	該当なし	bearer_resource_command
69	該当なし	該当なし	bearer_resource_failure_indication
70	該当なし	ran_info_relay	downlink_failure_indication
71	該当なし	該当なし	trace_session_activation
72	該当なし	該当なし	trace_session_deactivation
73	該当なし	該当なし	stop_paging_indication
95	該当なし	該当なし	create_bearer_request
96	該当なし	mbms_notification_request	create_bearer_response
97	該当なし	mbms_notification_response	update_bearer_request
98	該当なし	mbms_notification_reject_request	update_bearer_response
99	該当なし	mbms_notification_reject_response	delete_bearer_request
100	該当なし	create_mbms_context_request	delete_bearer_response
101	該当なし	create_mbms_context_response	delete_pdn_request
102	該当なし	update_mbms_context_request	delete_pdn_response
103	該当なし	update_mbms_context_response	該当なし
104	該当なし	delete_mbms_context_request	該当なし
105	該当なし	delete_mbms_context_response	該当なし

値	バージョン 0	バージョン 1	バージョン 2
112	該当なし	mbms_register_request	該当なし
113	該当なし	mbms_register_response	該当なし
114	該当なし	mbms_deregister_request	該当なし
115	該当なし	mbms_deregister_response	該当なし
116	該当なし	mbms_session_start_request	該当なし
117	該当なし	mbms_session_start_response	該当なし
118	該当なし	mbms_session_stop_request	該当なし
119	該当なし	mbms_session_stop_response	該当なし
120	該当なし	mbms_session_update_request	該当なし
121	該当なし	mbms_session_update_response	該当なし
128	該当なし	ms_info_change_request	identification_request
129	該当なし	ms_info_change_response	identification_response
130	該当なし	該当なし	sgsn_context_request
131	該当なし	該当なし	sgsn_context_response
132	該当なし	該当なし	sgsn_context_ack
133	該当なし	該当なし	forward_relocation_request
134	該当なし	該当なし	forward_relocation_response
135	該当なし	該当なし	forward_relocation_complete
136	該当なし	該当なし	forward_relocation_complete_ack
137	該当なし	該当なし	forward_access
138	該当なし	該当なし	forward_access_ack
139	該当なし	該当なし	relocation_cancel_request
140	該当なし	該当なし	relocation_cancel_response
141	該当なし	該当なし	configuration_transfer_tunnel
149	該当なし	該当なし	detach
150	該当なし	該当なし	detach_ack

値	バージョン 0	バージョン 1	バージョン 2
151	該当なし	該当なし	cs_paging
152	該当なし	該当なし	ran_info_relay
153	該当なし	該当なし	alert_mme
154	該当なし	該当なし	alert_mme_ack
155	該当なし	該当なし	ue_activity
156	該当なし	該当なし	ue_activity_ack
160	該当なし	該当なし	create_forward_tunnel_request
161	該当なし	該当なし	create_forward_tunnel_response
162	該当なし	該当なし	suspend
163	該当なし	該当なし	suspend_ack
164	該当なし	該当なし	resume
165	該当なし	該当なし	resume_ack
166	該当なし	該当なし	create_indirect_forward_tunnel_request
167	該当なし	該当なし	create_indirect_forward_tunnel_response
168	該当なし	該当なし	delete_indirect_forward_tunnel_request
169	該当なし	該当なし	delete_indirect_forward_tunnel_response
170	該当なし	該当なし	release_access_bearer_request
171	該当なし	該当なし	release_access_bearer_response
176	該当なし	該当なし	downlink_data
177	該当なし	該当なし	downlink_data_ack
179	該当なし	該当なし	pgw_restart
180	該当なし	該当なし	pgw_restart_ack
200	該当なし	該当なし	update_pdn_request
201	該当なし	該当なし	update_pdn_response
211	該当なし	該当なし	modify_access_bearer_request
212	該当なし	該当なし	modify_access_bearer_response

値	バージョン 0	バージョン 1	バージョン 2
231	該当なし	該当なし	mbms_session_start_request
232	該当なし	該当なし	mbms_session_start_response
233	該当なし	該当なし	mbms_session_update_request
234	該当なし	該当なし	mbms_session_update_response
235	該当なし	該当なし	mbms_session_stop_request
236	該当なし	該当なし	mbms_session_stop_response
240	data_record_transfer_request	data_record_transfer_request	該当なし
241	data_record_transfer_response	data_record_transfer_response	該当なし
254	該当なし	end_marker	該当なし
255	pdu	pdu	該当なし

gtp_info キーワード

1 つの GTP メッセージには多数の情報要素が含まれることがあり、それぞれの要素は定義済み数値および定義済み文字列によって識別されます。`gtp_info` キーワードを使用すると、指定された情報要素の先頭から検査を開始し、検査対象を指定の情報要素に限定することができます。定義されているメッセージ タイプと情報要素は GTP バージョンによって異なるため、このキーワードを使用するときには、`gtp_version` も使用する必要があります。

情報要素に対して定義された 10 進数値と定義された文字列のどちらでも指定できます。単一の値または文字列を指定することも、1 つのルール内で複数の `gtp_info` キーワードを使って複数の情報要素を検査することもできます。

1 つのメッセージに同じタイプの複数の情報要素が含まれている場合は、すべてが照合対象として検査されます。情報要素が無効な順序で出現する場合は、最後のインスタンスだけが検査されます。

GTP バージョンに応じて、同じ情報要素の値が異なる場合があることに注意してください。たとえば `cause` 情報要素の値は GTPv0 と GTPv1 では 1 ですが、GTPv2 では 2 です。

パケット内のバージョン番号に応じて、`gtp_info` キーワードは異なる値と一致します。上記の例の場合、GTPv0 または GTPv1 パケットではキーワードが情報要素値 1 と一致しますが、GTPv2 パケットでは値 2 と一致します。パケット内の情報要素値が、パケットで指定されたバージョンの既知の値でない場合は、キーワードがパケットと一致しません。

情報要素に整数を指定した場合、パケット内で指定されたバージョンとは無関係に、キーワード内のメッセージ タイプが GTP パケット内の値と一致すればキーワードが一致します。

次の表に、GTP 情報要素ごとにシステムで認識される値と文字列を示します。

表 *55 : GTP* 情報要素

値	バージョン 0	バージョン 1	バージョン 2
1	cause	cause	imsi
2	imsi	imsi	cause
3	rai	rai	recovery
4	tlli	tlli	該当なし
5	p_tmsi	p_tmsi	該当なし
[6]	qos	該当なし	該当なし
8	recording_required	recording_required	該当なし
9	認証	認証	該当なし
11	map_cause	map_cause	該当なし
12	p_tmsi_sig	p_tmsi_sig	該当なし
13	ms_validated	ms_validated	該当なし
18	recovery	recovery	該当なし
15	selection_mode	selection_mode	該当なし
16	flow_label_data_1	teid_1	該当なし
17	flow_label_signalling	teid_control	該当なし
18	flow_label_data_2	teid_2	該当なし
19	ms_unreachable	teardown_ind	該当なし
20	該当なし	nsapi	該当なし
21	該当なし	ranap	該当なし
22	該当なし	rab_context	該当なし
23	該当なし	radio_priority_sms	該当なし
24	該当なし	radio_priority	該当なし
25	該当なし	packet_flow_id	該当なし
26	該当なし	charging_char	該当なし
27	該当なし	trace_ref	該当なし

値	バージョン 0	バージョン 1	バージョン 2
36	該当なし	trace_type	該当なし
29	該当なし	ms_unreachable	該当なし
71	該当なし	該当なし	apn
72	該当なし	該当なし	ambr
73	該当なし	該当なし	ebi
74	該当なし	該当なし	ip_addr
75	該当なし	該当なし	mei
76	該当なし	該当なし	msisdn
77	該当なし	該当なし	indication
78	該当なし	該当なし	pco
79	該当なし	該当なし	paa
80	該当なし	該当なし	bearer_qos
80	該当なし	該当なし	flow_qos
82	該当なし	該当なし	rat_type
83	該当なし	該当なし	serving_network
84	該当なし	該当なし	bearer_tft
85	該当なし	該当なし	tad
86	該当なし	該当なし	uli
87	該当なし	該当なし	f_teid
88	該当なし	該当なし	tmsi
89	該当なし	該当なし	cn_id
90	該当なし	該当なし	s103pdf
91	該当なし	該当なし	s1udf
92	該当なし	該当なし	delay_value
93	該当なし	該当なし	bearer_context
94	該当なし	該当なし	charging_id

値	バージョン 0	バージョン 1	バージョン 2
95	該当なし	該当なし	charging_char
96	該当なし	該当なし	trace_info
97	該当なし	該当なし	bearer_flag
99	該当なし	該当なし	pdn_type
100	該当なし	該当なし	pti
101	該当なし	該当なし	drx_parameter
103	該当なし	該当なし	gsm_key_tri
104	該当なし	該当なし	umts_key_cipher_quin
105	該当なし	該当なし	gsm_key_cipher_quin
106	該当なし	該当なし	umts_key_quin
107	該当なし	該当なし	eps_quad
108	該当なし	該当なし	umts_key_quad_quin
109	該当なし	該当なし	pdn_connection
110	該当なし	該当なし	pdn_number
111	該当なし	該当なし	p_tmsi
112	該当なし	該当なし	p_tmsi_sig
113	該当なし	該当なし	hop_counter
114	該当なし	該当なし	ue_time_zone
115	該当なし	該当なし	trace_ref
116	該当なし	該当なし	complete_request_msg
117	該当なし	該当なし	guti
118	該当なし	該当なし	f_container
119	該当なし	該当なし	f_cause
120	該当なし	該当なし	plmn_id
121	該当なし	該当なし	target_id
123	該当なし	該当なし	packet_flow_id

値	バージョン 0	バージョン 1	バージョン 2
124	該当なし	該当なし	rab_contex
125	該当なし	該当なし	src_rnc_pdcp
126	該当なし	該当なし	udp_src_port
127	charge_id	charge_id	apn_restriction
128	end_user_address	end_user_address	selection_mode
129	mm_context	mm_context	src_id
130	pdp_context	pdp_context	該当なし
131	apn	apn	change_report_action
132	protocol_config	protocol_config	fq_csid
133	gsn	gsn	channel
134	msisdn	msisdn	emlpp_pri
135	該当なし	qos	node_type
136	該当なし	authentication_qu	fqdn
137	該当なし	tft	ti
138	該当なし	target_id	mbms_session_duration
139	該当なし	utran_trans	mbms_service_area
140	該当なし	rab_setup	mbms_session_id
141	該当なし	ext_header	mbms_flow_id
142	該当なし	trigger_id	mbms_ip_multicast
143	該当なし	omc_id	mbms_distribution_ack
144	該当なし	ran_trans	rfsp_index
145	該当なし	pdp_context_pri	uci
146	該当なし	addi_rab_setup	csg_info
147	該当なし	sgsn_number	csg_id
148	該当なし	common_flag	cmi
149	該当なし	apn_restriction	service_indicator

値	バージョン 0	バージョン 1	バージョン 2
150	該当なし	radio_priority_lcs	detach_type
151	該当なし	rat_type	ldn
152	該当なし	user_loc_info	node_feature
153	該当なし	ms_time_zone	mbms_time_to_transfer
154	該当なし	imei_sv	throttling
155	該当なし	camel	arp
156	該当なし	mbms_ue_context	epc_timer
157	該当なし	tmp_mobile_group_id	signalling_priority_indication
158	該当なし	rim_routing_addr	tmgi
159	該当なし	mbms_config	mm_srvcc
160	該当なし	mbms_service_area	flags_srvcc
161	該当なし	src_rnc_pdcp	nmbr
162	該当なし	addi_trace_info	該当なし
163	該当なし	hop_counter	該当なし
164	該当なし	plmn_id	該当なし
165	該当なし	mbms_session_id	該当なし
166	該当なし	mbms_2g3g_indicator	該当なし
167	該当なし	enhanced_nsapi	該当なし
168	該当なし	mbms_session_duration	該当なし
169	該当なし	addi_mbms_trace_info	該当なし
170	該当なし	mbms_session_repetition_num	該当なし
171	該当なし	mbms_time_to_data	該当なし
173	該当なし	bss	該当なし
174	該当なし	cell_id	該当なし
175	該当なし	pdu_num	該当なし
177	該当なし	mbms_bearer_capab	該当なし

値	バージョン 0	バージョン 1	バージョン 2
178	該当なし	rim_routing_disc	該当なし
179	該当なし	list_pfc	該当なし
180	該当なし	ps_xid	該当なし
181	該当なし	ms_info_change_report	該当なし
182	該当なし	direct_tunnel_flags	該当なし
183	該当なし	correlation_id	該当なし
184	該当なし	bearer_control_mode	該当なし
185	該当なし	mbms_flow_id	該当なし
186	該当なし	mbms_ip_multicast	該当なし
187	該当なし	mbms_distribution_ack	該当なし
188	該当なし	reliable_inter_rat_handover	該当なし
189	該当なし	rfsp_index	該当なし
190	該当なし	fqdn	該当なし
191	該当なし	evolved_allocation1	該当なし
192	該当なし	evolved_allocation2	該当なし
193	該当なし	extended_flags	該当なし
194	該当なし	uci	該当なし
195	該当なし	csg_info	該当なし
196	該当なし	csg_id	該当なし
197	該当なし	cmi	該当なし
198	該当なし	apn_ambr	該当なし
199	該当なし	ue_network	該当なし
200	該当なし	ue_ambr	該当なし
201	該当なし	apn_ambr_nsapi	該当なし
202	該当なし	ggsn_backoff_timer	該当なし
203	該当なし	signalling_priority_indication	該当なし

値	バージョン 0	バージョン 1	バージョン 2
204	該当なし	signalling_priority_indication_nsapi	該当なし
205	該当なし	high_bitrate	該当なし
206	該当なし	max_mbr	該当なし
251	charging_gateway_addr	charging_gateway_addr	該当なし
255	private_extension	private_extension	private_extension

SCADA キーワード

ルールエンジンは Modbus、DNP3、CIP、および S7Commplus のルールを使用して特定のプロトコルフィールドにアクセスします。

Modbus キーワード

Modbus キーワードを単独で使用することも、`content` や `byte_jump` など他のキーワードと組み合わせて使用することもできます。

modbus_data

`modbus_data` キーワードを使用すると、Modbus 要求または応答内の [Data] フィールドの先頭を指し示すことができます。

modbus_func

`modbus_func` キーワードを使用すると、Modbus アプリケーション層要求または応答見出し内の [Function Code（機能コード）] フィールドを照合できます。Modbus 機能コードとして、1 つの定義済み 10 進数値または 1 つの定義済み文字列を指定できます。

次の表に、Modbus機能コードとしてシステムで認識される定義済みの値と文字列を示します。

表 *56 : Modbus* 機能コード

値	文字列
1	read_coils
2	read_discrete_inputs
3	read_holding_registers
4	read_input_registers
5	write_single_coil
[6]	write_single_register

値	文字列
7	read_exception_status
8	diagnostics
11	get_comm_event_counter
12	get_comm_event_log
15	write_multiple_coils
16	write_multiple_registers
17	report_slave_id
20	read_file_record
21	write_file_record
22	mask_write_register
23	read_write_multiple_registers
24	read_fifo_queue
43	encapsulated_interface_transport

modbus_unit

`modbus_unit` キーワードを使用すると、Modbus 要求または応答ヘッダー内の [Unit ID] フィールドで 1 つの 10 進数値を照合できます。

DNP3 キーワード

DNP3 キーワードを単独で使用することも、`content` や `byte_jump` など他のキーワードと組み合わせて使用することもできます。

dnp3_data

`dnp3_data` キーワードを使用すると、再構築された DNP3 アプリケーション層フラグメントの先頭を指し示すことができます。

DNP3 プリプロセッサは、リンク層フレームをアプリケーション層フラグメントに再構築します。`dnp3_data` キーワードは、各アプリケーション層フラグメントの先頭を指し示します。他のルール オプションは、16 バイトごとにデータを分離してチェックサムを追加せずに、フラグメント内の再構築されたデータを照合することができます。

dnp3_func

`dnp3_func` キーワードを使用すると、DNP3 アプリケーション層要求または応答ヘッダー内の [機能コード（Function Code）] フィールドを照合できます。DNP3 機能コードとして、1 つの定義済み 10 進数値または 1 つの定義済み文字列を指定できます。

次の表に、DNP3 機能コードとしてシステムで認識される定義済みの値と文字列を示します。

表 *57 : DNP3* 機能コード

値	文字列
[0]	confirm
1	read
2	write
3	select
4	operate
5	direct_operate
[6]	direct_operate_nr
7	immed_freeze
8	immed_freeze_nr
9	freeze_clear
10	freeze_clear_nr
11	freeze_at_time
12	freeze_at_time_nr
13	cold_restart
18	warm_restart
15	initialize_data
16	initialize_appl
17	start_appl
18	stop_appl
19	save_config
20	enable_unsolicited
21	disable_unsolicited
22	assign_class
23	delay_measure
24	record_current_time
25	open_file

値	文字列
26	close_file
27	delete_file
36	get_file_info
29	authenticate_file
30	abort_file
31	activate_config
32	authenticate_req
33	authenticate_err
129	response
130	unsolicited_response
131	authenticate_resp

dnp3_ind

`dnp3_ind` キーワードを使用すると、DNP3 アプリケーション層応答ヘッダー内の [Internal Indications] フィールド内のフラグを照合できます。

1 つの既知のフラグ、または次の例のようなカンマ区切りのフラグ リストを示す文字列を指定できます。

```
class_1_events, class_2_events
```

複数のフラグを指定した場合、キーワードはリスト内の任意のフラグと一致します。いくつかのフラグの組み合わせを検出するには、1 つのルール内で `dnp3_ind` キーワードを複数回使用します。

定義済みの DNP3 内部通知フラグとしてシステムによって認識される文字列構文を以下に示します。

```
class_1_events
class_2_events
class_3_events
need_time
local_control
device_trouble
device_restart
no_func_code_support
object_unknown
parameter_error
event_buffer_overflow
already_executing
config_corrupt
reserved_2
reserved_1
```

dnp3_obj

dnp3_obj キーワードを使用すると、要求または応答内の DNP3 オブジェクト ヘッダーを照合できます。

DNP3データは、アナログ入力やバイナリ入力など、さまざまなタイプの一連のDNP3オブジェクトで構成されます。各タイプは、それぞれ 10 進数値で識別されるグループを使って区別されます（アナログ入力グループ、バイナリ入力グループなど）。各グループ内のオブジェクトは、それぞれオブジェクト データ形式を指定するオブジェクト バリエーションによってさらに区別されます（16 ビット整数、32 ビット整数、短精度浮動小数点など）。また、オブジェクト バリエーションの各タイプは 10 進数値でも識別可能です。

オブジェクト ヘッダーを識別する際には、オブジェクト ヘッダー グループのタイプを示す 10 進数値とオブジェクト バリエーションのタイプを示す 10 進数値を指定します。この 2 つの組み合わせによって DNP3 オブジェクトの特定のタイプが定義されます。

CIP および ENIP のキーワード

次のキーワードを単体でまたは組み合わせて使用すると、CIP プリプロセッサで検出された CIP および ENIP トラフィックに対する攻撃を識別するカスタム侵入ルールを作成できます。設定可能なキーワードについては、許容範囲内の単一の整数を指定します。詳細については、CIP プリプロセッサを参照してください。

表 *58*：

対象キーワード	照合先	範囲
cip_attribute	CIP メッセージの [オブジェクト クラス/インスタンス属性（Object Class/Instance Attribute）] フィールド。定義された 1 つの整数値を指定します。	0 ～ 65535
cip_class	CIP メッセージの [オブジェクト クラス（Object Class）] フィールド。定義された 1 つの整数値を指定します。	0 ～ 65535
cip_conn_path_class	接続パスのオブジェクト クラス。1 つの整数値を指定します。	0 ～ 65535
cip_instance	CIP メッセージの [インスタンス ID（Instance ID）] フィールド。1 つの整数値を指定します。	0 ～ 4284927295
cip_req	サービス要求メッセージ。	該当なし
cip_rsp	サービス応答メッセージ。	該当なし
cip_service	CIP サービス要求メッセージの [サービス（Service）] フィールド。1 つの整数値を指定します。	0 ～ 127

対象キーワード	照合先	範囲
cip_status	CIP サービス応答メッセージの [ステータス（Status）] フィールド。1 つの整数値を指定します。	0 ～ 255
enip_command	EthNet/IP ヘッダーのコマンド コード。1 つの整数値を指定します。	0 ～ 65535
enip_req	EthNet/IP 要求メッセージ。	該当なし
enip_rsp	EthNet/IP 応答メッセージ。	該当なし

S7Commplus キーワード

S7Commplusプリプロセッサが検出したトラフィックに対する攻撃を識別するカスタム侵入ルールを作成するには、S7Commplus キーワードを単体で使用するか、または組み合わせて使用します。設定可能なキーワードについては、許容範囲内の既知の単一の値か、または単一の整数を指定します。詳細については、S7Commplus プリプロセッサを参照してください。

次の点に注意してください。

- 同じルール内の複数の S7commplus キーワードは、AND 演算されます。
- 同じルールで複数の s7commplus_func キーワードまたは s7commplus_opcode キーワードを使用すると、ルールが無効になり、トラフィックに一致しなくなります。これらのキーワードで複数の値を検索するには、複数のルールを作成します。

s7commplus_content

S7Commplus侵入ルールで content キーワードまたは protected_content キーワードを使用する前に、s7commplus_content キーワードを使用して、カーソルをパケットペイロードの先頭に配置します。詳細については、content キーワードと protected_content キーワード （28 ページ）を参照してください。

s7commplus_func

s7commplus_func キーワードを使用して、S7Commplus ヘッダー内の次の値の 1 つと照合します。

- explore
- createobject
- deleteobject
- setvariable
- getlink
- setmultivar

- getmultivar
- beginsequence
- endsequence
- invoke
- getvarsubstr
- 0x0 ～ 0xFFF

  数式では追加の値を使用できることに注意してください。

s7commplus_opcode

`s7commplus_opcode` キーワードを使用して、S7Commplus ヘッダー内の次の値の 1 つと照合します。

- request
- response
- 通知
- response2
- 0x0 ～ 0xFF

  数式では追加の値を使用できることに注意してください。

パケット特性

特定のパケット特性を持つパケットに対してのみイベントを生成するルールを作成できます。

dsize

`dsize` キーワードはパケット ペイロード サイズを検査します。「大なり」演算子と「小なり」演算子（<、>）を使って値の範囲を指定することができます。次の構文をに従って範囲を指定できます。

```
>number_of_bytes
<number_of_bytes
number_of_bytes<>number_of_bytes
```

たとえば、400 バイトを超えるパケット サイズを指定するには、`dtype` 値として `>400` を使用します。500 バイト未満のパケット サイズを指定するには、`<500` を使用します。400 ～ 500 バイトのパケットに対してルールをトリガーとして使用するよう指定するには、`400<>500` を使用します。

⚠

注意 `dsize` キーワードは、プリプロセッサによってデコードされる前のパケットを検査します。

isdataat

isdataat キーワードは、ペイロード内の特定の位置にデータが存在することを確認するよう、ルール エンジンに指示します。

次の表に、isdataat キーワードで使用可能な引数を列挙します。

表 *59 : isdataat* の引数

引数	タイプ	説明
オフセット（Offset）	必須	ペイロード内の特定の位置。たとえば、パケット ペイロード内のバイト位置 50 にデータが出現することを検査するには、オフセット値として 50 を指定します。! 修飾子は isdataat 検査の結果を否定します。特定量のデータがペイロードに存在しない場合は警告が出されます。 また、既存の byte_extract 変数または byte_math 結果を使用してこの引数の値を指定することもできます。
相対的（Relative）	オプション	最後に見つかったコンテンツ一致を基準にして相対的な位置を計算します。相対位置を指定する場合は、カウンタがバイト 0 から始まることに注意してください。最後に見つかったコンテンツ一致から順方向に移動するバイト数から 1 を差し引いて位置を計算します。たとえば、最後に見つかったコンテンツ一致から 9 バイト後にデータが出現すべきことを指定するには、相対オフセットとして 8 を指定します。
raw データ（Raw Data）	オプション	Firepower システム プリプロセッサによるデコードやアプリケーション層の正規化が行われる前の、元のパケット ペイロードにデータが配置されていることを指定します。前のコンテンツ一致が未加工パケット データ内に存在していた場合は、この引数を **Relative** と一緒に使用できます。

たとえば、foo というコンテンツを検索するルールで isdataat の値が次のように指定される場合、

- Offset = !10
- Relative = enabled

ルール エンジンが foo の後ろからペイロード末尾までに 10 バイトを検出しない場合、システムは警告を出します。

sameip

sameip キーワードは、パケットの送信元 IP アドレスと宛先 IP アドレスが同じであることを検査します。このキーワードは引数を受け入れません。

fragoffset

`fragoffset` キーワードは、フラグメント化されたパケットのオフセットを検査します。一部の exploit（WinNuke サービス拒否攻撃など）では、特定のオフセットを持つ手動生成されたパケット フラグメントが使われるため、このキーワードが役立ちます。

たとえば、フラグメント化されたパケットのオフセットが31337バイトかどうかを検査するには、`fragoffset` 値として `31337` を指定します。

`fragoffset` キーワードの引数を指定するときには、次の演算子を使用できます。

表 *60 : fragoffset* キーワードの引数演算子

演算子	説明
!	ノット
>	より大きい
<	より少ない

否定（!）演算子を < や > と組み合わせて使用できないことに注意してください。

cvs

`cvs` キーワードは、Concurrent Versions System（CVS）トラフィック内で不正な形式の CVS エントリを検査します。攻撃者は不正な形式のエントリを使用して、ヒープ オーバーフローを強制的に発生させ、CVS サーバ上で有害コードを実行することができます。このキーワードを使用すると、2 つの既知の CVS 脆弱性 CVE-2004-0396（CVS 1.11.x ～ 1.11.15 と 1.12.x ～ 1.12.7）および CVS-2004-0414（CVS 1.12.x ～ 1.12.8 と 1.11.x ～ 1.11.16）に対する攻撃を識別できます。`cvs` キーワードは、正しい形式のエントリであることを検査して、不正な形式のエントリが検出された場合はアラートを生成します。

CVS が動作するポートをルールに含める必要があります。さらに、トラフィックが発生する可能性のあるポートを TCP ポリシー内のストリーム再構築用のポートリストに追加することで、CVS セッションの状態を保持できるようにする必要があります。ストリーム再構築が行われるクライアント ポートのリストには、TCP ポート 2401（`pserver`）と 514（`rsh`）が含まれています。ただし、サーバが `xinetd` サーバ（つまり pserver）として動作する場合は、任意の TCP ポート上で動作できることに注意してください。すべての非標準ポートを、ストリーム再構築の [クライアント ポート（Client Ports）] リストに追加します。

関連トピック

byte_extract キーワード （48 ページ）
TCP ストリームのプリプロセス オプション

アクティブ応答のキーワード

resp キーワードおよび **react** キーワードにより、2 つの方法でアクティブ応答を開始できます。パケットがどちらかのキーワードを含む侵入ルールをトリガーとして使用すると、その侵入

ルールにより、1つのアクティブ応答が開始します。アクティブ応答のキーワードは、トリガーとして使用された TCP ルールに応答して TCP 接続を閉じるために、またはトリガーとして使用された UDP ルールに応答して UDP セッションを閉じるために、アクティブ応答を開始します。侵入廃棄ルールでのアクティブ応答を参照してください。（攻撃者がアクティブ応答を無視または回避するよう選択する可能性があるなど）さまざまな理由で、アクティブ応答はファイアウォールの代わりとして想定されていません。

アクティブ応答は、ルーテッド展開またはトランスペアレント展開を含むインライン展開でサポートされます。たとえば、インライン展開での `react` キーワードに応答して、システムは接続の両端用のトラフィックに TCP リセット（RST）パケットを直接挿入でき、通常はこれによって接続が閉じます。アクティブ応答は、パッシブ展開ではサポートされていないか、または適していません。

アクティブ応答は戻って来ることがあるため、システムは TCP リセットによる TCP リセットの開始を許可しません。これにより、アクティブ応答が無限に続くことを防止できます。また、システムは、標準的な慣行に従って ICMP 到達不能パケットによる ICMP 到達不能パケットの開始を許可しません。

侵入ルールがアクティブ応答をトリガーとして使用した後、TCP 接続で追加のトラフィックを検出するよう、TCP ストリーム プリプロセッサを設定できます。追加のトラフィックが検出されると、プリプロセッサは、指定された最大値まで、追加のアクティブ応答を接続またはセッションの両端に送信します。トランスポート/ネットワーク プリプロセッサの詳細オプションの「**アクティブ応答の最大数（Maximum Active Responses）**」および「**最小応答時間（秒）（Minimum Response Seconds）**」を参照してください。

関連トピック

侵入廃棄ルールでのアクティブ応答

resp キーワード

`resp` キーワードを使用すると、ルール ヘッダーで TCP プロトコルと UDP プロトコルのどちらが指定されているかに基づいて、TCP 接続または UDP セッションにアクティブに（能動的に）応答できます。

キーワード引数を使用すると、パケットの方向、および TCP リセット（RST）パケットと ICMP 到達不能パケットのどちらをアクティブ応答として使用するかを指定できます。

任意の TCP リセット引数または ICMP 到達不能引数を使用して、TCP 接続を閉じることができます。UDP セッションを閉じるには、ICMP 到達不能引数だけを使用する必要があります。

また、さまざまな TCP リセット引数を使用することで、パケットの送信元、宛先、またはその両方にアクティブ応答を送ることができます。すべての ICMP 到達不能引数はパケット送信元に送られます。ICMP ネットワーク、ホスト、またはポートのどの到達不能パケットを使用するか（または 3 つすべてを使用するか）を指定できます。

ルールがトリガーとして使用されたときにシステムで実行されるアクションを正確に指定するために、`resp` キーワードで使用できる引数を次の表に列挙します。

表 *61 : resp* 引数

引数	説明
reset_source	ルールをトリガーとして使用したパケットを送信元エンドポイントに TCP リセット パケットを送ります。この代わりに、下位互換性のためにサポートされている `rst_snd` を指定することもできます。
reset_dest	ルールをトリガーとして使用したパケットの宛先であるエンドポイントに TCP リセット パケットを送ります。この代わりに、下位互換性のためにサポートされている `rst_rcv` を指定することもできます。
reset_both	送信側エンドポイントと受信側エンドポイントの両方に TCP リセット パケットを送ります。この代わりに、下位互換性のためにサポートされている `rst_all` を指定することもできます。
icmp_net	送信側に ICMP ネットワーク到達不能メッセージを送ります。
icmp_host	送信側に ICMP ホスト到達不能メッセージを送ります。
icmp_port	送信側に ICMP ポート到達不能メッセージを送ります。この引数は、UDP トラフィックを終了するために使われます。
icmp_all	送信側に次の ICMP メッセージを転送します。 • ネットワーク到達不能 • ホスト到達不能 • ポート到達不能

たとえば、ルールがトリガーとして使用されたときに接続の両側をリセットするようルールを設定するには、`resp` キーワードの値として `reset_both` を使用します。

次のように、カンマ区切りのリストを使用して複数の引数を指定できます。

```
argument,argument,argument
```

react キーワード

`react` キーワードを使用すると、パケットがルールをトリガーとして使用した時点でデフォルト HTML ページを TCP 接続クライアントに送信できます。HTML ページの送信後に、システムは TCP リセット パケットを使って接続の両端へのアクティブ応答を開始します。`react` キーワードは UDP トラフィックのアクティブ応答をトリガーとして使用しません。

オプションで、次の引数を指定できます。

```
msg
```

`msg` 引数を使用する `react` ルールがパケットによってトリガーとして使用されると、HTML ページにルール イベント メッセージが表示されます。

msg 引数を指定しない場合、HTML ページには次のメッセージが含まれます。

```
You are attempting to access a forbidden site.
Consult your system administrator for details.
```

（注） アクティブ応答は戻されることがあるため、HTML 応答ページによって react ルールがトリガーとして使用されないようにしてください（結果としてアクティブ応答が無限に続く可能性があります）。Cisco では、react ルールを十分にテストしてから実稼動環境でアクティブにするよう推奨しています。

関連トピック

ルールの詳細 （2 ページ）

detection_filter キーワード

detection_filter キーワードを使用すると、指定された時間内に指定された数のパケットがルールをトリガーとして使用しない限り、ルールでイベントが生成されないようにすることができます。これにより、早すぎるタイミングでルールがイベントを生成することを回避できます。たとえば、数秒間にログイン試行が 2 ～ 3 回失敗することは想定の範囲内ですが、同じ時間内に多数の試行が発生した場合はブルートフォースアタックを示唆している可能性があります。

detection_filter キーワードの必須の引数は、送信元/宛先のどちらの IP アドレスをシステムで追跡するか、イベントをトリガーすめたに検出基準が満たされるべき回数、およびカウントの継続時間を定義します。

イベントのトリガーを遅らせるには、次の構文を使用します。

```
track by_src/by_dst, count count, seconds number_of_seconds
```

track 引数は、ルールの検出基準を満たすパケット数をカウントするときに、パケットの送信元 IP アドレスと宛先 IP アドレスのどちらを使用するかを指定します。システムでイベントインスタンスを追跡する方法を指定するには、次の表の中から引数値を選択します。

表 *62 : detection_filter* の追跡引数

引数	説明
by_src	送信元 IP アドレスによる検出基準カウント。
by_dst	宛先 IP アドレスによる検出基準カウント。

count 引数は、ルールでイベントを生成するために、指定された時間内に指定された IP アドレスのルールをトリガーすべきパケットの数を指定します。

seconds 引数は、ルールでイベントを生成するために、指定された数のパケットがルールをトリガーすべき時間枠を秒数で指定します。

パケット内でコンテンツ foo を検索するルールが、次の引数を含む detection_filter キーワードを使用するとします。

```
track by_src, count 10, seconds 20
```

この例のルールは、特定の送信元 IP アドレスから 20 秒以内に 10 個のパケットで foo を検出するまでは、イベントを生成しません。システムが最初の 20 秒以内に foo を含むパケットを 7 つしか検出しなかった場合は、イベントが生成されません。しかし、最初の 20 秒間で foo が 40 回出現した場合は、ルールで 30 個のイベントが生成され、20 秒が経過するとカウントが再開されます。

しきい値と detection_filter キーワードの比較

detection_filter キーワードは、非推奨の threshold キーワードに代わるものです。threshold キーワードは、下位互換性を維持するために引き続きサポートされていますが、侵入ポリシー内で設定されるしきい値と同じ機能です。

detection_filter キーワードは、パケットがルールをトリガーとして使用する前に適用される検出機能です。ルールは、指定されたパケット カウントの前に検出されたトリガー パケットに関してイベントを生成しません。また、インライン展開では、パケットを破棄するようルールで設定されていても、そのようなパケットを破棄しません。逆に、指定されたパケット カウントの後に出現する、ルールをトリガーとして使用するパケットに関してルールはイベントを生成します。また、インライン展開でパケットを破棄するよう設定されている場合は、そのようなパケットを破棄します。

しきい値は、検出アクションを発生させないイベント通知機能です。これは、パケットがイベントをトリガーとして使用した後に適用されます。インライン展開において、パケットを破棄するよう設定されたルールは、ルールしきい値とは無関係に、ルールをトリガーとして使用するすべてのパケットを破棄します。

侵入ポリシー内で detection_filter キーワードを侵入イベントしきい値、侵入イベント抑制、および Rate-Based 攻撃防御機能と任意に組み合わせて使用できることに注意してください。また、侵入ポリシー内の侵入イベントしきい値機能と組み合わせて非推奨の threshold キーワードを使用するインポートされたローカル ルールを有効にした場合、ポリシー検証が失敗することに注意してください。

関連トピック

侵入イベントしきい値
侵入ポリシー抑制の設定
[ルール（Rule）] ページからの動的ルール状態の設定

tag キーワード

ホストまたはセッションに関する追加のトラフィックをログに記録するようシステムに指示するには、tag キーワードを使用します。tag キーワードを使って検出するトラフィックのタイプと量を指定するときには、次の構文を使用します。

```
tagging_type, count, metric, optional_direction
```

次の 3 つの表に、その他の使用可能な引数について説明します。

2 つのタイプのタグ機能から選択できます。次の表に、これらのタグ機能の説明を示します。侵入ルールでルール ヘッダー オプションのみを設定した場合、session タグ引数タイプによって、同じセッションからのパケットが別のセッションからのパケットのように記録されることに注意してください。同じセッションからのパケットをまとめてグループ化するには、同じ侵入ルール内で 1 つ以上のルール オプション（`flag` キーワードや `content` キーワードなど）を設定します。

表 *63 : tag* の引数

引数	説明
session	ルールをトリガーとして使用したセッション内のパケットをログに記録します。
host	ルールをトリガーとして使用したパケットを送信したホストからのパケットをログに記録します。ホストからのトラフィックのみ（`src`）、またはホストへのトラフィックのみ（`dst`）を記録する方向修飾子を追加できます。

ログに記録するトラフィック量を指定するには、次の引数を使用します。

表 *64 :* カウント引数

引数	説明
count	ルールがトリガーとして使用された後にログに記録するパケット数または秒数。 この単位を指定するには、count 引数の後に測定基準引数を使用します。

次の表の中から、トラフィックの時間または量ごとにログで使用する測定基準を選択してください。

注意 高帯域ネットワークでは、1 秒あたり数千パケットが発生する可能性があり、多数のパケットにタグを付けるとパフォーマンスに重大な影響が及ぶ可能性があるため、必ずネットワーク環境に合わせてこの設定を調整してください。

表 *65 :* ログの測定基準引数

引数	説明
packets	ルールのトリガー後に、カウントで指定されるパケット数をログに記録します。
seconds	ルールのトリガー後に、カウントで指定される秒数の間、トラフィックを記録します。

たとえば、次の tag キーワード値を使用するルールがトリガーとして使用された場合、

```
host, 30, seconds, dst
```

次の30秒間にクライアントからホストに送信されるすべてのパケットがログに記録されます。

flowbits キーワード

状態名をセッションに割り当てるには、flowbits キーワードを使用します。すでに名前が付けられた状態に基づいてセッション内の後続パケットを分析することにより、システムは単一セッション内で複数のパケットに及ぶエクスプロイトを検出して警告を出すことができます。

flowbits 状態名は、セッションの特定部分でパケットに割り当てられるユーザー定義のラベルです。パケットの内容に基づいてパケットに状態名を付けると、警告の必要のないパケットと有害なパケットを区別しやすくなります。管理対象デバイスごとに最大 1024 個の状態名を定義できます。たとえば、ログイン成功後にのみ発生することがわかっている有害パケットについて警告するには、flowbits キーワードを使用して、初期ログイン試行を構成するパケットを除去することにより、有害パケットに焦点を絞ることができます。このような機能を実装するには、まず、セッション内のすべてのログイン確立済みパケットに logged_in 状態のラベルを付けるルールを作成した後、2 番目のルールを作成し、最初のルールで設定された状態を持つパケットを検査してそのようなパケットだけを処理する flowbits をそのルールに含めます。

オプションの *group name* を使用すると、状態のグループに状態名を含めることができます。1 つの状態名は複数のグループに属することができます。グループに関連付けられていない状態は相互排他的ではないため、トリガーとして使用されたルールがグループに関連付けられていない状態を設定した場合、現在設定されている他の状態には影響がありません。

flowbits キーワードのオプション

次の表に、flowbits キーワードで使用できる演算子、状態、およびグループのさまざまな組み合わせについて説明します。なお、状態名には、英数字、ピリオド（.）、アンダースコア（_）、およびダッシュ（-）を含めることができます。

表 *66 : flowbits* のオプション

演算子	状態オプション	グループ	説明
set	state_name	オプション	パケットに関する指定された状態を設定します。グループが定義されている場合は、指定されたグループ内で状態を設定します。
set	state_name&state_name	オプション	パケットに関する指定された状態を設定します。グループが定義されている場合は、指定されたグループ内で状態を設定します。
setx	state_name	入力必須	指定されたグループ内でパケットに関して指定された状態を設定し、グループ内の他のすべての状態を解除します。

演算子	状態オプション	グループ	説明
setx	state_name&state_name	入力必須	指定されたグループ内でパケットに関して指定された状態を設定し、グループ内の他のすべての状態を解除します。
unset	state_name	グループなし	パケットに関する指定された状態を解除します。
unset	state_name&state_name	グループなし	パケットに関する指定された状態を解除します。
unset	all	入力必須	指定されたグループ内のすべての状態を解除します。
toggle	state_name	グループなし	指定された状態が設定されている場合はそれを解除し、指定された状態が解除されている場合にはそれを設定します。
toggle	state_name&state_name	グループなし	指定された複数の状態が設定されている場合はそれらを解除し、指定された複数の状態が解除されている場合はそれらを設定します。
toggle	all	入力必須	指定されたグループ内で設定されているすべての状態を解除し、指定されたグループ内で解除されているすべての状態を設定します。
isset	state_name	グループなし	指定された状態がパケット内で設定されているかどうかを判別します。
isset	state_name&state_name	グループなし	指定された複数の状態がパケット内で設定されているかどうかを判別します。
isset	state_name|state_name	グループなし	指定されたいずれかの状態がパケット内で設定されているかどうかを判別します。
isset	any	入力必須	指定されたグループ内で、いずれかの状態が設定されているかどうかを判別します。
isset	all	入力必須	指定されたグループ内で、すべての状態が設定されているかどうかを判別します。
isnotset	state_name	グループなし	指定された状態がパケット内で設定されていないかどうかを判別します。
isnotset	state_name&state_name	グループなし	指定された複数の状態がパケット内で設定されていないかどうかを判別します。
isnotset	state_name|state_name	グループなし	指定されたいずれかの状態が、パケット内で設定されていないかどうかを判別します。

演算子	状態オプション	グループ	説明
isnotset	any	入力必須	パケット内でいずれかの状態が設定されていないかどうかを判別します。
isnotset	all	入力必須	パケット内ですべての状態が設定されていないかどうかを判別します。
reset	（状態なし）	オプション	すべてのパケットのすべての状態を解除します。グループが指定されている場合、グループ内のすべての状態を解除します。
noalert	（状態なし）	グループなし	イベント生成を抑制するには、これを他の演算子と組み合わせて使用します。

flowbits キーワードの使用に関するガイドライン

flowbits キーワードを使用するときには、次の点に注意してください。

- setx 演算子を使用する場合、指定した状態は、指定したグループ以外のグループに属することができません。
- setx 演算子を複数回定義して、それぞれのインスタンスで別々の状態と同じグループを指定できます。
- setx 演算子を使用してグループを指定する場合、そのグループに対して set、toggle、unset 演算子を使用することはできません。
- isset 演算子と isnotset 演算子は、指定された状態がグループに含まれるかどうかに関係なく、その状態を評価します。
- 侵入ポリシーの保存時、侵入ポリシーの再適用時、および（アクセス コントロール ポリシーで参照される侵入ポリシー数に関係なく）アクセス コントロール ポリシーの適用時には、グループ**指定のない** isset または isnotset 演算子を含むルールを有効にした場合、対応する状態名とプロトコルに関する flowbits 割り当て（set、setx、unset、toggle）に影響する 1 つ以上のルールを有効にしないと、対応する状態名の flowbits 割り当てに影響するすべてのルールが有効になります。
- 侵入ポリシーの保存時、侵入ポリシーの再適用時、および（アクセス コントロール ポリシーで参照される侵入ポリシー数に関係なく）アクセス コントロール ポリシーの適用時には、グループを**指定した** isset 演算子または isnotset 演算子を含むルールを有効にした場合、flowbits 割り当て（set、setx、unset、toggle）に影響し、対応するグループ名を定義するすべてのルールもまた有効になります。

flowbits キーワードの例

この項では、flowbits キーワードを使用する 3 つの例を示します。

flowbits キーワードの例：state_name を使用した設定

これは、`state_name` を使用した flowbits 設定の例です。

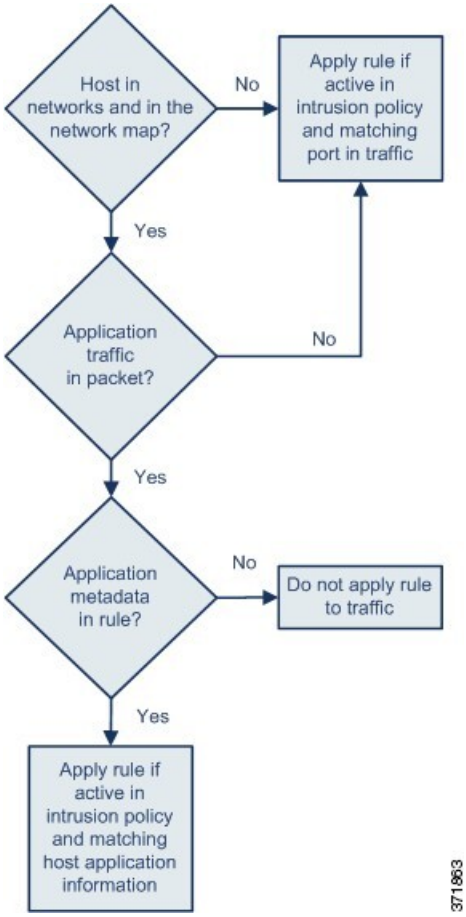
CVE ID 2000-0284 に記述されている IMAP 脆弱性について考えてみます。この脆弱性は、IMAP の実装（具体的には LIST、LSUB、RENAME、FIND、および COPY コマンド）で見られます。ただし、攻撃者がこの脆弱性を悪用するには、IMAP サーバにログインする必要があります。IMAP サーバからの LOGIN 確認とそれに続く exploit は必然的に別々のパケットに存在するため、この exploit を検出する非フロー ベースのルールを作成するのは困難です。`flowbits` キーワードを使って一連のルールを作成すると、ユーザが IMAP サーバにログイン済みかどうかを追跡し、ログイン済みの場合は、いずれかの攻撃が検出された時点でイベントを生成できます。ユーザがログイン済みでない場合、攻撃によって脆弱性が悪用されることはないため、イベントが生成されません。

下記の 2 つのルール フラグメントはこの例を示しています。最初のルール フラグメントは IMAP サーバーからの IMAP ログイン確認を検索します。

```
alert tcp any 143 -> any any (msg:"IMAP login"; content:"OK

LOGIN"; flowbits:set,logged_in; flowbits:noalert;)
```

次の図は、上記のルール フラグメントにおける `flowbits` キーワードの効果を示しています。

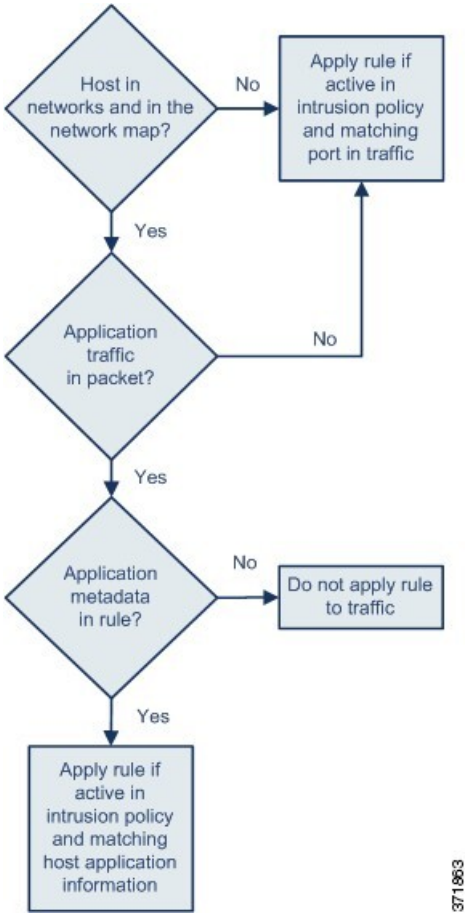


`flowbits:set` は `logged_in` 状態を設定しますが、`flowbits:noalert` がアラートを抑制することに注意してください。これは、IMAP サーバー上で多数の無害なログインセッションが見つかる可能性があるためです。

次のルール フラグメントは LIST 文字列を検索しますが、セッション内の先行パケットの結果として logged_in 状態が設定済みでない限り、イベントを生成しません。

```
alert tcp any any -> any 143 (msg:"IMAP LIST";
content:"LIST"; flowbits:isset,logged_in;)
```

次の図は、上記のルール フラグメントにおける `flowbits` キーワードの効果を示しています。



この場合、最初のフラグメントを含むルールが先行パケットによってトリガーとして使用した場合、2 番目のフラグメントを含むルールがトリガーとして使用し、イベントを生成します。

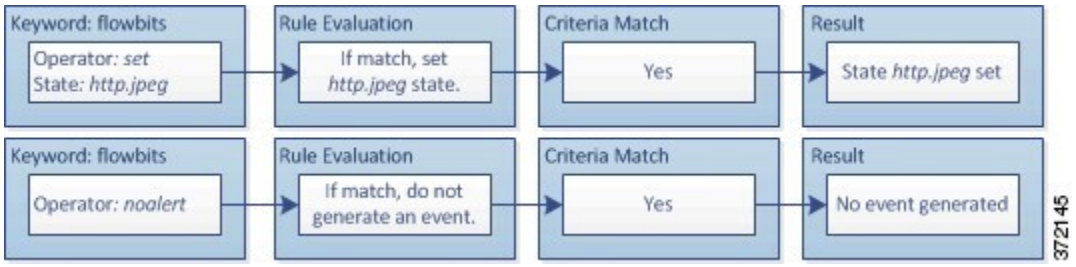
flowbits キーワードの例：誤検出イベントを引き起こす設定

後続パケット内コンテンツが、効力を失った状態を持つルールに一致することによって誤検出イベントが発生する可能性があります。複数のルールで設定された複数の状態名をグループに含めることでこれを回避できます。次の例は、複数の状態名をグループに含めない場合に誤検出が発生する可能性があることを示しています。

1 つのセッションで次の 3 つのルール フラグメントがこの順序でトリガーとして使用される場合を考えてみます。

```
(msg:"JPEG transfer";
content:"image/";pcre:"/^Content-?Type\x3a(\s*|\s*\r?\n\s+)image\x2fp?jpe?g/smi";
?flowbits:set,http.jpeg; flowbits:noalert;)
```

次の図は、上記のルール フラグメントにおける `flowbits` キーワードの効果を示しています。

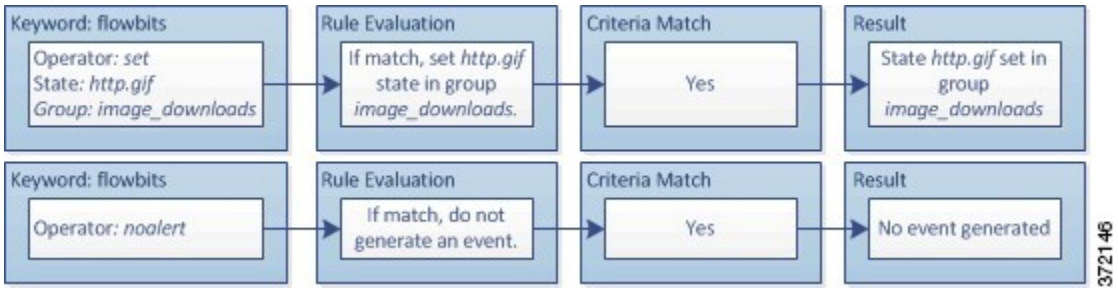


最初のルール フラグメント内の `content` キーワードと `pcre` キーワードが JPEG ファイル ダウンロードに一致し、`flowbits:set,http.jpeg` が `http.jpeg flowbits` ステートを設定し、`flowbits:noalert` はルールでのイベント生成を抑制します。イベントが生成されない理由は、このルールの目的がファイル ダウンロードを検出して `flowbits` 状態を設定することだからです。これにより、1 つ以上のコンパニオン ルールで状態名を検査して有害コンテンツを探し、有害コンテンツが検出された時点でイベントを生成できます。

次のルール フラグメントは、上記の JPEG ファイル ダウンロードに続く GIF ファイル ダウンロードを検出します。

```
(msg:"GIF transfer"; content:"image/";
pcre:"/^Content-?Type\x3a(\s*|\s*\r?\n\s+)image\x2fgif/smi";
?flowbits:set,http.jpg,image_downloads; flowbits:noalert;)
```

次の図は、上記のルール フラグメントにおける `flowbits` キーワードの効果を示しています。

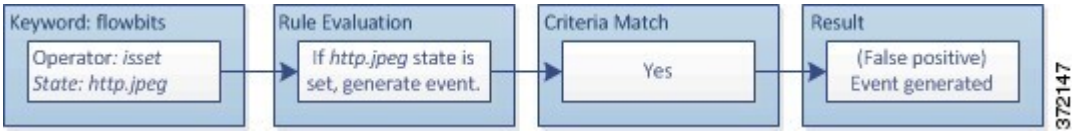


2 番目のルール内の `content` キーワードと `pcre` キーワードは GIF ファイル ダウンロードを照合し、`flowbits:set,http.jpg` は `http.jpg` flowbit ステートを設定し、`flowbits:noalert` はルールでのイベント生成を抑制します。最初のルール フラグメントで設定された `http.jpeg` 状態が不要になっても引き続き設定されていることに注意してください。これは、後続の GIF ダウンロードが検出されたときに JPEG ダウンロードが既に終了しているはずであるためです。

次に示す 3 番目のルール フラグメントは最初のルール フラグメントのコンパニオンです。

```
(msg:"JPEG exploit";?flowbits:isset,http.jpeg;content:"|FF|";
pcre:"?/\xFF[\xE1\xE2\xED\xFE]\x00[\x00\x01]/";)
```

次の図は、上記のルール フラグメントにおける flowbits キーワードの効果を示しています。



3 番目のルール フラグメントでは、もはや無意味になった http.jpeg ステートが設定されていることを flowbits:isset,http.jpeg が判別し、content と pcre は（GIF ファイルでは無害でも）JPEG ファイル内では有害とみなされるコンテンツを照合します。3 番目のルール フラグメントによって、JPEG ファイル内に存在しないエクスプロイトに関する誤検出イベントが生成されます。

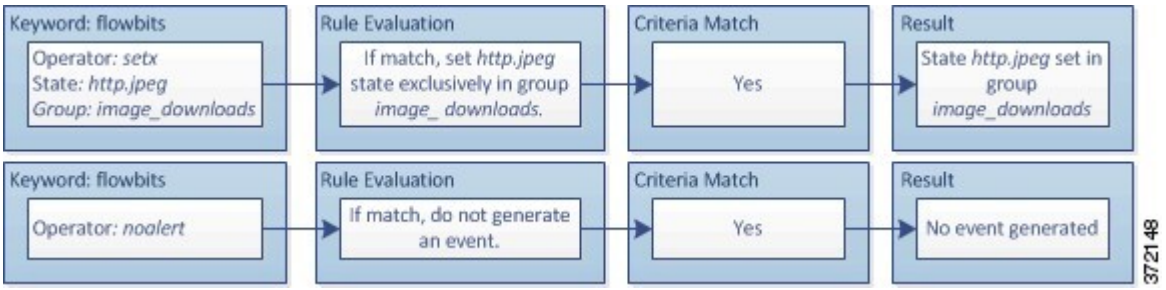
flowbits キーワードの例：誤検出イベントを防ぐための設定

次の例は、状態名をグループに含めて setx 演算子を使用することで、どのように誤検出を防止できるかを示しています。

前の例とほぼ同じケースを考えます。ただし、最初の 2 つのルールで、同じ状態グループに 2 つの異なる状態名が含まれるようになった点が異なります。

```
(msg:"JPEG transfer";
content:"image/";pcre:"/^Content-?Type\x3a(\s*|\s*\r?\n\s+)image\x2fp?jpe?g/smi";
?flowbits:setx,http.jpeg,image_downloads; flowbits:noalert;)
```

次の図は、上記のルール フラグメントにおける flowbits キーワードの効果を示しています。

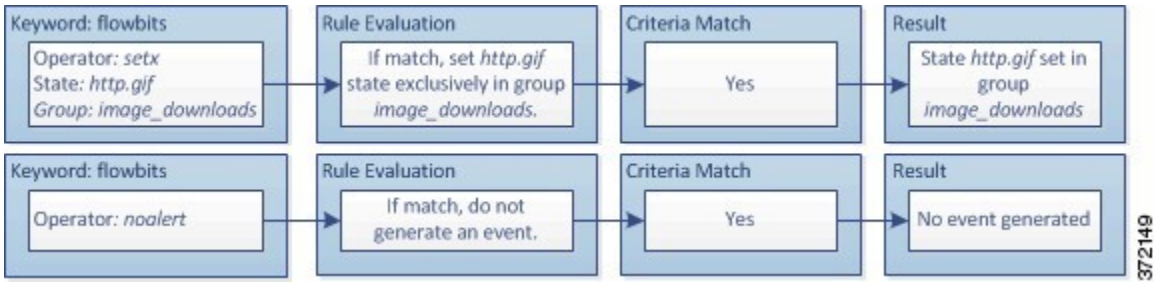


最初のルール フラグメントが JPEG ファイル ダウンロードを検出すると、flowbits:setx,http.jpeg,image_downloads キーワードが flowbits 状態を http.jpeg に設定し、その状態を image_downloads グループに含めます。

その後、次のルールが後続の GIF ファイル ダウンロードを検出します。

```
(msg:"GIF transfer"; content:"image/";
pcre:"/^Content-?Type\x3a(\s*|\s*\r?\n\s+)image\x2fgif/smi";
?flowbits:setx,http.jpg,image_downloads; flowbits:noalert;)
```

次の図は、上記のルール フラグメントにおける flowbits キーワードの効果を示しています。

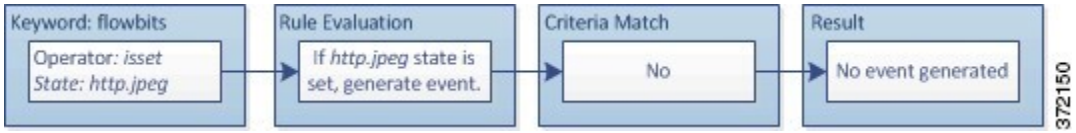


2 番目のルール フラグメントが GIF ダウンロードに一致すると、`flowbits:setx,http.jpg,image_downloads` キーワードが `http.jpg flowbits` ステートを設定し、グループ内の他のステートである `http.jpeg` を解除します。

次に示す 3 番目のルール フラグメントで誤検出は発生しません。

```
(msg:"JPEG exploit"; ?flowbits:isset,http.jpeg;content:"|FF|";
pcre:"/?\xFF[\xE1\xE2\xED\xFE]\x00[\x00\x01]/";)
```

次の図は、上記のルール フラグメントにおける `flowbits` キーワードの効果を示しています。



`flowbits:isset,http.jpeg` が false であるため、ルール エンジンはルールの処理を停止し、イベントは生成されません。こうして、GIF ファイル内のコンテンツが JPEG ファイルに関するエクスプロイト コンテンツと一致した場合でも誤検出が回避されます。

http_encode キーワード

`http_encode` キーワードを使用すると、HTTP URI、HTTP ヘッダー内の非 cookie データ、HTTP 要求ヘッダー内の cookie、HTTP 応答内の set-cookie データのいずれかにおいて、正規化前の HTTP 要求または応答内のエンコード タイプに基づいてイベントを生成できます。

HTTP 応答と HTTP cookie を検査し、`http_encode` キーワードを使用しているルールに一致したものを返すように、HTTP Inspect プリプロセッサを設定する必要があります。

また、侵入ルール内の `http_encode` キーワードで特定のエンコード タイプによってイベントがトリガーとして使用されるようにするには、HTTP Inspect プリプロセッサ設定で個々の特定のエンコード タイプのデコード オプションとアラート オプションの両方を有効にする必要があります。

次の表は、このオプションでイベントを生成できる、HTTP URI、ヘッダー、cookie、set-cookie のエンコード タイプを説明しています。

表 *67 : http_encode* エンコード タイプ

エンコード タイプ	説明
utf8	HTTP Inspect プリプロセッサによるデコードで UTF-8 エンコード タイプが有効になっている場合、指定された場所で UTF-8 エンコードを検出します。
double_encode	HTTP Inspect プリプロセッサによるデコードで二重エンコード タイプが有効になっている場合、指定された場所で二重エンコードを検出します。
non_ascii	非 ASCII 文字が検出されても、検出されたエンコード タイプが有効になっていない場合に、指定された場所で非 ASCII 文字を検出します。
uencode	HTTP Inspect プリプロセッサによるデコードで Microsoft %u エンコード タイプが有効になっている場合、指定された場所で Microsoft %u エンコードを検出します。
bare_byte	HTTP Inspect プリプロセッサによるデコードで空白バイト エンコード タイプが有効になっている場合、指定された場所で空白バイト エンコードを検出します。

関連トピック

HTTP Inspect プリプロセッサ
サーバーレベルの HTTP 正規化オプション

http_encode キーワードの構文

エンコーディングの場所

HTTP URI、ヘッダー、または set-cookie などの Cookie で指定されたエンコーディング タイプを検索するかどうかを指定します。

エンコード タイプ

次のいずれかの形式を使用して、1 つ以上のエンコード タイプを指定します。

```
encode_type
encode_type|encode_type|encode_type...
```

ここで、`encode_type` は次のいずれかです。

```
utf8
double_encode
non_ascii
uencode
bare_byte.
```

否定（!）演算子と OR（|）演算子を一緒に使用できないことに注意してください。

http_encode キーワードの例：2 つの http_endcode キーワードを使用した 2 つのエンコーディングの検索

次に、同じルールで 2 つの `http_encode` キーワードを使用して、UTF-8 および Microsoft IIS %u エンコーディングの HTTP URI を検索する例を示します。

最初に、`http_encode` キーワードを使用します。

- **エンコーディングの場所**：`HTTP URI`
- **エンコーディングのタイプ**：`utf8`

次に、追加の `http_encode` キーワードを使用します。

- **エンコーディングの場所**：`HTTP URI`
- **エンコーディングのタイプ**：`uencode`

概要：file_type および file_group キーワード

`file_type` と `file_group` キーワードを使用すると、タイプとバージョンに基づいて、FTP、HTTP、SMTP、IMAP、POP3、NetBIOS-ssn（SMB）を介して伝送されるファイルを検出できます。1 つの侵入ルール内で複数の `file_type` キーワードや `file_group` キーワードを使用**しない**でください。

ヒント 脆弱性データベース（VDB）を更新すると、最新のファイル タイプ、バージョン、グループが侵入ルール エディタに表示されます。

（注） システムは、`file_type` および `file_group` キーワードに値を代入するためにプリプロセッサを自動的に有効にすることはしません。

`file_type` または `file_group` キーワードに一致するトラフィックに対してイベントを生成し、インライン展開では、違反パケットをドロップします。するには、特定のプリプロセッサを有効にする**必要があります**。

表 *68 : file_type* および *file_group* の侵入イベントの生成

プロトコル	必要なプリプロセッサまたはプリプロセッサ オプション
FTP	FTP/Telnet プリプロセッサおよび [TCP ペイロードの正規化（Normalize TCP Payload）] インライン正規化プリプロセッサ オプション
HTTP	HTTP トラフィックでの侵入イベントを生成する HTTP Inspect プリプロセッサ。

プロトコル	必要なプリプロセッサまたはプリプロセッサ オプション
SMTP	HTTP トラフィックでの侵入イベントを生成する SMTP プリプロセッサ
IMAP	IMAP プリプロセッサ
POP3	POP プリプロセッサ
NetBIOS-ssn（SMB）	DCE/RPC プリプロセッサおよび [SMB ファイル インスペクション（SMB File Inspection）] DCE/RPC プリプロセッサ オプション

関連トピック

FTP/Telnet デコーダ
インライン正規化プリプロセッサ
HTTP Inspect プリプロセッサ
SMTP プリプロセッサ
IMAP プリプロセッサ
POP プリプロセッサ
DCE/RPC プリプロセッサ

file_type キーワードと file_group キーワード

file_type

`file_type` キーワードを使用すると、トラフィック内で検出対象となるファイルのタイプとバージョンを指定できます。ファイル タイプ引数（**JPEG** や **PDF** など）は、トラフィックで検出するファイルの形式を識別します。

（注） 同じ侵入ルール内で `file_type` キーワードを別の `file_type` キーワードまたは `file_group` キーワードと一緒に**使用しないでください**。

デフォルトでは **Any Version** が選択されますが、一部のファイル タイプではバージョン オプション（たとえば PDF バージョン **1.7**）を選択することにより、トラフィックで検出対象となる特定のファイル タイプ バージョンを識別できます。

file_group

`file_group` キーワードを使用すると、トラフィック内で検出する類似のファイル タイプからなる Cisco 定義のグループを選択できます（**マルチメディア**、**オーディオ**など）。また、ファイル グループには、グループ内の各ファイル タイプに関する Cisco 定義のバージョンも含まれています。

（注） 同じ侵入ルール内で file_group キーワードを別の file_group キーワードまたは file_type キーワードと一緒に使用**しない**でください。

file_data キーワード

file_data キーワードは、content、byte_jump、byte_test、pcre などの他のキーワードで使用可能な位置引数の参照として機能するポインタです。file_data キーワードが指し示すデータのタイプは、検出されるトラフィックによって決まります。file_data キーワードを使用すると、次のペイロード タイプの先頭を指し示すことができます。

- HTTP 応答本文

  HTTP 応答パケットを検査するには、HTTP Inspect プリプロセッサを有効にして、HTTP 応答を検査するようプリプロセッサを設定する必要があります。HTTP Inspect プリプロセッサが HTTP 応答本文データを検出した場合に、file_data キーワードが一致します。

- 非圧縮 gzip ファイル データ

  HTTP 応答本文内の非圧縮 gzip ファイルを検査するには、HTTP Inspect プリプロセッサを有効にする必要があり、さらに HTTP 応答を検査して HTTP 応答本文内の gzip 圧縮ファイルを復元するようプリプロセッサを設定する必要があります。詳細については、サーバレベルの HTTP 正規化オプション [HTTP 応答の検査（Inspect HTTP Responses）] および [圧縮データの検査（Inspect Compressed Data）] を参照してください。file_data キーワードは、HTTP Inspect プリプロセッサが HTTP 応答本文内で非圧縮 gzip データを検出した場合に一致します。

- 正規化された JavaScript

  正規化された JavaScript データを検査するには、HTTP Inspect プリプロセッサを有効にして、HTTP 応答を検査するようプリプロセッサを設定する必要があります。file_data キーワードは、HTTP Inspect プリプロセッサが応答本文データ内で JavaScript を検出した場合に一致します。

- SMTP ペイロード

  SMTP ペイロードを検査するには、SMTP プリプロセッサを有効にする必要があります。file_data キーワードは、SMTP プリプロセッサが SMTP データを検出した場合に一致します。

- SMTP、POP、または IMAP トラフィック内のエンコードされた電子メール添付ファイル

  SMTP、POP、または IMAP トラフィック内の電子メール添付ファイルを検査するには、それぞれ SMTP、POP、または IMAP プリプロセッサを単独で、または任意に組み合わせて有効にする必要があります。その後、有効にしたプリプロセッサごとに、デコード対象のそれぞれの添付ファイル エンコード タイプをデコードするようプリプロセッサが設定されていることを確認する必要があります。プリプロセッサごとに設定可能な添付ファイ

ル デコード オプションは、[Base64 復号の深さ（Base64 Decoding Depth）]、[7 ビット/8 ビット/バイナリ復号の深さ（7-Bit/8-Bit/Binary Decoding Depth）]、[Quoted Printable 復号の深さ（Quoted-Printable Decoding Depth）]、および [UNIX 間復号の深さ（Unix-to-Unix Decoding Depth）] です。

1 つのルール内で複数の `file_data` キーワードを使用できます。

関連トピック

HTTP Inspect プリプロセッサ
サーバーレベルの HTTP 正規化オプション
SMTP プリプロセッサ
IMAP プリプロセッサ

pkt_data キーワード

`pkt_data` キーワードは、`content`、`byte_jump`、`byte_test`、`pcre` などの他のキーワードで使用可能な位置引数の参照として機能するポインタです。

正規化された FTP、Telnet、または SMTP トラフィックが検出された場合、`pkt_data` キーワードは、正規化されたパケット ペイロードの先頭を指します。その他のトラフィックが検出された場合、`pkt_data` キーワードは、未加工の TCP または UDP ペイロードの先頭を指します。

侵入ルールで検査するために、該当するトラフィックをシステムで正規化するには、次の正規化オプションを有効にする必要があります。

- 検査のために FTP トラフィックを正規化するには、FTP & Telnet プリプロセッサの [FTP コマンドでの Telnet エスケープ コードの検出（Detect Telnet Escape codes within FTP commands）] オプションを有効にします。
- 検査のために Telnet トラフィックを正規化するには、FTP & Telnet プリプロセッサの Telnet の [正規化（Normalize）] オプションを有効にします。
- 検査のために SMTP トラフィックを正規化するには、SMTP プリプロセッサの [正規化（Normalize）] オプションを有効にします。

1 つのルール内で複数の `pkt_data` キーワードを使用できます。

関連トピック

クライアントレベルの FTP オプション
Telnet オプション
SMTP プリプロセッサのオプション

base64_decode キーワードと base64_data キーワード

`base64_decode` キーワードと `base64_data` キーワードを組み合わせて使用すると、指定したデータを Base64 データとしてデコードおよび検査するようルール エンジンに指示できます。たと

えば HTTP PUT および POST 要求内の Base64 エンコード HTTP 認証要求見出しと Base64 エンコード データを検査する場合に、これが役立つ可能性があります。

これらのキーワードは特に、HTTP 要求内の Base64 データをデコードして検査するうえで役立ちます。また、長いヘッダー行を複数行に拡張するために HTTP で使われるのと同じ方法でスペース文字やタブ文字を使用する SMTP などのプロトコルでも、これらを使用できます。この行拡張（折り返しとも言う）を使用するプロトコル内に行拡張が存在しない場合、後続スペース/タブを伴わない復帰または改行が出現した箇所で検査が終了します。

base64_decode

`base64_decode` キーワードは、パケット データを Base64 データとしてデコードするようルール エンジンに指示します。オプションの引数を使用すると、デコードするバイト数と、デコードを開始するデータ内の位置を指定できます。

`base64_decode` キーワードは 1 つのルール内で 1 回だけ使用可能です。また、少なくとも 1 つの `base64_data` キーワードのインスタンスの前にこれを配置する必要があります。

Base64 データをデコードする前に、ルール エンジンは、複数行にわたって折り返された長いヘッダーを元どおりに広げます。ルールエンジンが次のいずれかに遭遇するとデコードが終了します。

- ヘッダー行の末尾
- デコード対象として指定されたバイト数
- パケットの末尾

次の表に、`base64_decode` キーワードで使用可能な引数の説明を示します。

表 *69 : base64_decode* のオプション引数

引数	説明
Bytes	デコードするバイト数を指定します。これを指定しない場合、ヘッダー行の末尾またはパケット ペイロード末尾のどちらかが先に出現するまでデコードが続行されます。ゼロ以外の正の値を指定できます。
Offset	パケット ペイロードの先頭を基準にしたオフセットを決定します。さらに **Relative** も指定した場合は、現在の検査位置を基準にしたオフセットを決定します。ゼロ以外の正の値を指定できます。
Relative	現在の検査位置を基準にして検査することを指定します。

base64_data

`base64_data` キーワードは、`base64_decode` キーワードを使ってデコードされた Base64 データを検査するための参照を提供します。`base64_data` キーワードは、デコードされた Base64 データの先頭から検査を開始するよう設定します。オプションで、`content` や `byte_test` などの他のキーワードで使用可能な位置引数を使用して、検査位置をさらに指定することもできます。

base64_decode キーワードを使用した後に base64_data キーワードを 1 回以上使用する必要があります。オプションで、base64_data を複数回使用して、デコードされた Base64 データの先頭に戻ることができます。

Base64 データを検査するときには、次の点に注意してください。

- 高速パターン マッチ機能は使用できません。
- 中間的な HTTP コンテンツ引数を使ってルール内で Base64 検査を中断する場合は、Base64 データをさらに検査する前に、別の base64_data キーワードをルールに挿入する必要があります。

関連トピック

概要：HTTP content および protected_content キーワードの引数 （34 ページ）
content キーワードの高速パターン マッチ機能の引数 （39 ページ）

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。