シグニチャの定義

この章では、シグニチャを定義および作成する方法について説明します。この章は、次の項で構成されています。

シグニチャについて

攻撃またはその他のネットワーク リソースの不正使用は、ネットワークへの侵入として定義付けることができます。シグニチャベースのテクノロジーを使用するセンサーは、ネットワークへの侵入を検出できます。*シグニチャ*は、DoS 攻撃（サービス拒絶攻撃）などの典型的な不正侵入行為を検出するためにセンサーが使用する規則の集まりです。センサーは、ネットワーク パケットをスキャンするときに、シグニチャを使って既知の攻撃を検出し、指定されたアクションに従って対応します。

センサーは、一連のシグニチャとネットワーク アクティビティを比較します。一致した場合、イベントのロギングやアラートの送信などのアクションを実行します。センサーでは、既存のシグニチャを変更したり、新しいシグニチャを定義したりできます。

シグニチャ ベースの侵入検出では、false positive が生じる場合があります。通常のネットワーク アクティビティでも、悪意のあるアクティビティとして誤解される場合があるためです。たとえば、ネットワーク アプリケーションやオペレーティング システムによっては多数の ICMP メッセージを送信する場合がありますが、シグニチャ ベースの検出システムでは、これを攻撃者がネットワーク セグメントを調査しようとしていると解釈してしまう可能性があります。センサーをチューニングすると、false positives を最小限に抑えることができます。

特定のシグニチャを使ってネットワーク トラフィックを監視するようにセンサーを設定するには、そのシグニチャを使用可能にする必要があります。デフォルトでは、重要なシグニチャはシグニチャ アップデートのインストール時に使用可能になります。使用可能になっているシグニチャと一致する攻撃を検出すると、センサーはアラートを生成します。アラートは、センサーのイベント ストアに保存されます。Web ベース クライアントは、アラートやその他のイベントをイベント ストアから取得できます。デフォルトでは、センサーは Informational 以上のすべてのアラートをログに記録します。

シグニチャには、サブシグニチャを持つもの（サブカテゴリに分類されているもの）があります。サブシグニチャを設定した場合、あるサブシグニチャのパラメータを変更しても、変更が適用されるのはそのサブシグニチャだけです。たとえば、シグニチャ 3050 のサブシグニチャ 1 の重大度を変更した場合、重大度の変更はサブシグニチャ 1 だけに適用され、3050 2、3050 3、および 3050 4 には適用されません。

IPS 5.1 には、1000 個を超えるデフォルトの組み込みシグニチャがあります。組み込みシグニチャのリストでシグニチャの名前を変更したり、シグニチャを削除したりすることはできません。ただし、シグニチャをリタイアさせ、センシング エンジンから除去することができます。リタイアにしたシグニチャは後でアクティブ化できます。ただし、このプロセスを実行すると、センシング エンジンは設定を再構築する必要があり時間がかかるため、トラフィック処理が遅れる可能性があります。組み込みシグニチャのチューニングはできます。これには、シグニチャのいくつかのパラメータを変更します。変更された組み込みシグニチャは、*チューニング済み*シグニチャと呼ばれます。

カスタム シグニチャと呼ばれるシグニチャを作成できます。カスタム シグニチャ ID は、60000 から始まります。これらは、UDP 接続におけるストリング照合、ネットワーク フラッドの追跡、および各種スキャンなどの多数の用途について設定できます。シグニチャは、監視するトラフィックの種類に対して特別に設計されたシグニチャ エンジンを使って作成します。

シグニチャ変数

この項では、シグニチャ変数について説明します。取り上げる事項は次のとおりです。

シグニチャ変数について

複数のシグニチャで同じ値を使用する場合、変数を使用します。変数の値を変更すると、すべてのシグニチャの変数が更新されます。このため、シグニチャを設定するときに変数を繰り返し変更しなくて済みます。

（注）　変数の前にドル記号($)を付けて、文字列ではなく変数を使用していることを示す必要があります。

シグニチャ システムに必要なため、削除できない変数もあります。変数が保護されている場合は、それを選択して編集することはできません。保護された変数を削除しようとすると、エラー メッセージが表示されます。編集できる変数は一度に 1 つだけです。

シグニチャ変数の設定

変数を作成するには、シグニチャ定義サブモードで **variables** コマンドを使用します。

次のオプションが適用されます。

- *variable-name*：この変数に割り当てる名前を定義します。
  名前には、数字または文字だけを使用できます。ハイフン（-）または下線（_）は使用できます。
- **ip-addr-range**：IP アドレスをグループ化するためのシステム定義変数。
  有効な値は、A.B.C.D-A.B.C.D[,A.B.C.D-A.B.C.D] です。
- **web-ports**：HTTP トラフィックを探すポート用のシステム定義変数。
  1 つの変数に対して複数のポート番号を指定するには、エントリをカンマで区切ります。たとえば、80, 3128, 8000, 8010, 8080, 8888, 24326 のように指定します。

シグニチャ変数を設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　IP アドレスのグループ用のシグニチャ変数を作成します。

```
sensor(config-sig)# variables IPADD ip-addr-range 10.1.1.1-10.1.1.24
```

ステップ 4　Web ポート用のシグニチャ変数を編集します。

```
sensor(config-sig)# variables WEBPORTS web-ports 80,3128,8000
```

WEBPORTS は Web サーバが実行されているポート群で、あらかじめ定義されているものですが、値は編集できます。この変数は、Web ポートが含まれるすべてのシグニチャに影響します。デフォルトは、80, 3128, 8000, 8010, 8080, 8888, 24326 です。

ステップ 5　変更を確認します。

```
sensor(config-sig)# show settings
   variables (min: 0, max: 256, current: 2)
   -----------------------------------------------
      variable-name: IPADD
      -----------------------------------------------
         ip-addr-range: 10.1.1.1-10.1.1.24
      -----------------------------------------------
      <protected entry>
      variable-name: WEBPORTS
      -----------------------------------------------
         web-ports: 80,3128,8000 default: 80-80,3128-3128,8000-8000,8010-8010,80
80-8080,8888-8888,24326-24326
      -----------------------------------------------
```

ステップ 6　シグニチャ定義サブモードを終了します。

```
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 7　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

シグニチャの設定

この項では、シグニチャのパラメータの設定方法について説明します。取り上げる事項は次のとおりです。

シグニチャの汎用パラメータの設定

特定のシグニチャの汎用パラメータの設定では、次のオプションが適用されます。

- **alert-frequency**：アラートをグループ化するためのサマリー オプションを設定します。

  手順については、P.7-6 の「アラート頻度の設定」を参照してください。
- **alert-severity**：アラートの重大度を設定します。

  手順については、P.7-8 の「アラート重大度の設定」を参照してください。
- **engine**：シグニチャ エンジンを指定します。エンジン サブモードでは、アクションを割り当てることができます。

  シグニチャ エンジンの詳細については、付録 B「シグニチャ エンジン」を参照してください。アクションを割り当てる手順については、P.7-13 の「シグニチャへのアクションの割り当て」を参照してください。
- **event-counter**：イベントのカウンタを設定します。

  手順については、P.7-9 の「イベント カウンタの設定」を参照してください。
- **promisc-delta**：アラートの重大度を決定するために使用されるデルタ値。

注意　シグニチャの promisc-delta 設定を変更することはお勧めしません。

混合モードでは、混合デルタによって特定のアラートの RR が低くなります。センサーはターゲット システムのアトリビュートを認識しませんが、混合モードではパケットを拒否できないため、混合アラートの優先順位を（優先順位の低い RR より）低く設定しておくと役立ちます。そうすることで、管理者は優先順位の高い RR アラートの調査に集中できます。

インライン モードでは、センサーが違反パケットを拒否することができます。その場合、違反パケットがターゲット ホストに到達することはないので、ターゲットが脆弱であっても問題になりません。攻撃はネットワーク上で許可されなかったため、RR 値は下げません。

サービス、OS、およびアプリケーションに固有のシグニチャ以外では、混合デルタは 0 です。シグニチャが OS、サービス、またはアプリケーションに固有のものである場合は、5、10、または 15 の混合デルタがカテゴリごとに 5 つのポイントから計算されます。

- **sig-description**：シグニチャの説明。

- **sig-fidelity-rating**：シグニチャの忠実度の評価。
  手順については、P.7-11 の「シグニチャ忠実度評価の設定」を参照してください。
- **status**：シグニチャのステータスをイネーブルまたはリタイアに設定します。
  手順については、P.7-12 の「シグニチャのステータスの設定」を参照してください。

アラート頻度の設定

シグニチャのアラート頻度を設定するには、シグニチャ定義サブモードで **alert-frequency** コマンドを使用します。

次のオプションが適用されます。

- *sig-id*：このシグニチャに割り当てられた固有の数値を示します。
  この値を使用すると、センサーが特定のシグニチャを識別できます。値は 1000 ～ 65000 です。
- *subsig-id*：このサブシグニチャに割り当てられた固有の数値を示します。
  サブシグニチャ ID は、広い範囲のシグニチャのバージョンをより細かく示すために使用します。値は 0 ～ 255 です。
- **alert-frequency**：シグニチャの反応中にセンサーがアラートを生成する頻度。
  次のパラメータをシグニチャに指定します。
  - **summary-mode**：センサーでアラートをグループ化する方法。
    fire-all：すべてのイベントについてアラートを生成します。
    fire-once：1 回だけアラートを生成します。
    global-summarize：攻撃者や被害先の数に関係なく 1 回だけアラートが生成されるようにアラートを要約します。
    summarize：すべてのアラートを要約します。
  - **summary-interval**：各サマリー アラートを生成する間隔（秒単位）。
    値は 1 ～ 65535 です。
  - **summary-key**：シグニチャの要約に使用されるストレージ タイプ。
    Axxx：攻撃者のアドレス。
    Axxb：攻撃者のアドレスと被害先のポート。
    AxBx：攻撃者と被害先のアドレス。
    AaBb：攻撃者と被害先のアドレスとポート。
    xxBx：被害先のアドレス。
  - **specify-global-summary-threshold [yes | no]**：グローバル サマリーしきい値（オプション）を設定するかどうかを指定します。
  - **global-summary-threshold**：イベント数のしきい値。この値を超えるとアラートはグローバル サマリーに要約されます。

シグニチャのアラート頻度パラメータを設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　設定するシグニチャを指定します。

```
sensor(config-sig)# signatures 9000 0
```

ステップ 4　アラート頻度サブモードに入ります。

```
sensor(config-sig-sig)# alert-frequency
```

ステップ 5　シグニチャのアラート頻度を設定します。

a.　たとえば、サマリー モードを fire-once に設定します。

```
sensor(config-sig-sig-ale)# summary-mode fire-once
sensor(config-sig-sig-ale-fir)# specify-global-summary-threshold yes
sensor(config-sig-sig-ale-fir-yes)# global-summary-threshold 3000
sensor(config-sig-sig-ale-fir-yes)# summary-interval 5000
```

b.　サマリー キーを設定します。

```
sensor(config-sig-sig-ale-fir-yes)# exit
sensor(config-sig-sig-ale-fir)# summary-key AxBx
```

c.　設定を確認します。

```
sensor(config-sig-sig-ale-fir)# show settings
   fire-once
   -----------------------------------------------
      summary-key: AxBx default: Axxx
      specify-global-summary-threshold
      -----------------------------------------------
         yes
         -----------------------------------------------
            global-summary-threshold: 3000 default: 120
            summary-interval: 5000 default: 15
         -----------------------------------------------
      -----------------------------------------------
   -----------------------------------------------
sensor(config-sig-sig-ale-fir)#
```

ステップ 6　アラート頻度サブモードを終了します。

```
sensor(config-sig-sig-ale-fir)# exit
sensor(config-sig-sig-ale)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 7　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

アラート重大度の設定

シグニチャの重大度を設定するには、シグニチャ定義サブモードで **alert-severity** コマンドを使用します。

次のオプションが適用されます。

- *sig-id*：このシグニチャに割り当てられた固有の数値を示します。
  この値を使用すると、センサーが特定のシグニチャを識別できます。値は 1000 ～ 65000 です。
- *subsig-id*：このサブシグニチャに割り当てられた固有の数値を示します。
  サブシグニチャ ID は、広い範囲のシグニチャのバージョンをより細かく示すために使用します。値は 0 ～ 255 です。
- **alert-severity**：アラートの重大度。
  - **high** ：危険なアラート。
  - **medium**：中レベルのアラート。
  - **low**：低レベルのアラート。
  - **informational**：情報アラート。
    これがデフォルトの方法です。

アラート重大度を設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　設定するシグニチャを選択します。

```
sensor(config-sig)# signatures 9000 0
```

ステップ 4　アラート重大度を割り当てます。

```
sensor(config-sig-sig)# alert-severity medium
```

ステップ 5　設定を確認します。

```
sensor(config-sig-sig)# show settings
   <protected entry>
   sig-id: 9000
   subsig-id: 0
   -----------------------------------------------
      alert-severity: medium default: informational
      sig-fidelity-rating: 75 <defaulted>
      promisc-delta: 0 <defaulted>
      sig-description
      ----------------------------------------------
         sig-name: Back Door Probe (TCP 12345) <defaulted>
         sig-string-info: SYN to TCP 12345 <defaulted>
         sig-comment: <defaulted>
         alert-traits: 0 <defaulted>
         release: 40 <defaulted>
      ----------------------------------------------
      engine
      ----------------------------------------------
         atomic-ip
         ---------------------------------------------
            event-action: produce-alert <defaulted>
            fragment-status: any <defaulted>
            specify-l4-protocol
            ---------------------------------------------
--MORE--
```

ステップ 6　シグニチャ サブモードを終了します。

```
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 7　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

イベント カウンタの設定

センサーでのイベントのカウント方法を設定するには、シグニチャ定義サブモードで **event-counter** コマンドを使用します。たとえば、同じシグニチャが同じアドレス セットに対して 5 回反応した場合にだけ、センサーがアラートを送信するように指定できます。

次のオプションが適用されます。

- **event-count**：アラートを生成するまでのイベントの発生回数。有効な範囲は 1 ～ 65535 です。デフォルトは 1 です。
- **event-count-key**：シグニチャのイベントをカウントするために使用するストレージ タイプ。
  - **Axxx**：攻撃者のアドレス。
  - **AxBx**：攻撃者と被害先のアドレス。
  - **Axxb**：攻撃者のアドレスと被害先のポート。
  - **xxBx**：被害先のアドレス。
  - **AaBb**：攻撃者と被害先のアドレスとポート。
- **specify-alert-interval [yes | no]**：アラートの間隔をイネーブルにします。
  - **alert-interval**：イベント カウントをリセットするまでの時間（秒単位）。デフォルトは 60 です。

イベント カウンタを設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　イベント カウンタを設定するシグニチャを選択します。

```
sensor(config-sig)# signatures 9000 0
```

ステップ 4　イベント カウンタ サブモードに入ります。

```
sensor(config-sig-sig)# event-counter
```

ステップ 5　アラートを生成するまでのイベントの発生回数を設定します。

```
sensor(config-sig-sig-eve)# event-count 2
```

ステップ 6　このシグニチャのイベントのカウントに使用するストレージ タイプを設定します。

```
sensor(config-sig-sig-eve)# event-count-key AxBx
```

ステップ 7　（オプション）アラート間隔をイネーブルにします。

```
sensor(config-sig-sig-eve)# specify-alert-interval yes
```

ステップ 8　（オプション）イベント カウントをリセットするまでの時間（秒単位）を指定します。

```
sensor(config-sig-sig-eve-yes)# alert-interval 30
```

ステップ 9　設定を確認します。

```
sensor(config-sig-sig-eve-yes)# exit
sensor(config-sig-sig-eve)# show settings
   event-counter
   -----------------------------------------------
      event-count: 2 default: 1
      event-count-key: AxBx default: Axxx
      specify-alert-interval
      -----------------------------------------------
         yes
         -----------------------------------------------
            alert-interval: 30 default: 60
         -----------------------------------------------
      -----------------------------------------------
   -----------------------------------------------
sensor(config-sig-sig-eve)#
```

ステップ 10 シグニチャ サブモードを終了します。

```
sensor(config-sig-sig-eve)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 11 変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

シグニチャ忠実度評価の設定

シグニチャのシグニチャ忠実度評価を設定するには、シグニチャ定義サブモードで **sig-fidelity-rating** コマンドを使用します。

次のオプションが適用されます。

- **sig-fidelity-rating**：対象とする特定の情報がない場合にこのシグニチャをどの程度忠実に実行するかに関連付ける重み値を指定します。

  有効な値は 0 ～ 100 です。

シグニチャのシグニチャ忠実度評価を設定するには、次の手順を実行します。

ステップ 1 管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2 シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3 設定するシグニチャを選択します。

```
sensor(config-sig)# signatures 12000 0
```

ステップ 4 シグニチャの忠実度評価を設定します。

```
sensor(config-sig-sig)# sig-fidelity-rating 50
```

ステップ 5　設定を確認します。

```
sensor(config-sig-sig)# show settings
   <protected entry>
   sig-id: 12000
   subsig-id: 0
   -----------------------------------------------
      alert-severity: low <defaulted>
      sig-fidelity-rating: 50 default: 85
      promisc-delta: 15 <defaulted>
      sig-description
      ----------------------------------------------
         sig-name: Gator Spyware Beacon <defaulted>
         sig-string-info: /download/ User-Agent: Gator <defaulted>
         sig-comment: <defaulted>
         alert-traits: 0 <defaulted>
         release: 71 <defaulted>
      ----------------------------------------------
```

ステップ 6　シグニチャ サブモードを終了します。

```
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 7　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

シグニチャのステータスの設定

特定のシグニチャのステータスを指定するには、シグニチャ定義モードで **status** コマンドを使用します。

次のオプションが適用されます。

- **status**：シグニチャがイネーブル、ディセーブル、リタイアのどの状態であるかを指定します。
  - **enabled [true | false]**：シグニチャをイネーブルにします。
  - **retired [true | false]**：シグニチャをリタイアにします。

注意　シグニチャのアクティブ化とリタイアには 30 分以上かかる場合があります。

シグニチャのステータスを変更するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　設定するシグニチャを選択します。

```
sensor(config-sig)# signatures 12000 0
```

ステップ 4　シグニチャのステータスを変更します。

```
sensor(config-sig-sig)# status
sensor(config-sig-sig-sta)# enabled true
```

ステップ 5　設定を確認します。

```
sensor(config-sig-sig-sta)# show settings
   status
   -----------------------------------------------
      enabled: true default: false
      retired: false <defaulted>
   -----------------------------------------------
sensor(config-sig-sig-sta)#
```

ステップ 6　シグニチャ サブモードを終了します。

```
sensor(config-sig-sig-sta)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 7　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

シグニチャへのアクションの割り当て

シグニチャが反応したときにセンサーが実行するアクションを設定するには、シグニチャ定義サブモードで **event-action** コマンドを使用します。

次のオプションが適用されます。

- **deny-attacker-inline**：（インライン モードのみ）指定された期間、攻撃者アドレスから、このパケットおよび将来のパケットを送信しません。
- **deny-attacker-service-pair-inline**：（インラインのみ）指定された期間、攻撃者アドレスと被害先のポートのペアで、このパケットおよび将来のパケットを送信しません。
- **deny-attacker-victim-pair-inline**：（インラインのみ）指定された期間、攻撃者 / 被害先のアドレスのペアで、このパケットおよび将来のパケットを送信しません。
- **deny-connection-inline**：（インライン モードのみ）TCP フローで、このパケットおよび将来のパケットを送信しません。
- **deny-packet-inline**：（インライン モードのみ）このパケットを送信しません。
- **log-attacker-packets**：攻撃者のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
- **log-pair-packets**：攻撃者と被害先のアドレスのペアを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。

- **log-victim-packets**：被害先のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
- **produce-alert**：イベントをアラートとしてイベント ストアに書き込みます。
- **produce-verbose-alert**：違反パケットの符号化ダンプ（切り捨てられている可能性があります）をアラートに組み込みます。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
- **request-block-connection**：この接続をブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
- **request-block-host**：この攻撃者ホストをブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
- **request-rate-limit**：レート制限を実行するレート制限要求を ARC に送信します。レート制限デバイスは、このアクションを実装するよう設定されている必要があります。
- **request-snmp-trap**：SNMP 通知を実行する要求をセンサーの通知アプリケーション コマンドに送信します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。SNMP は、このアクションを実装するようセンサーで設定されている必要があります。
- **reset-tcp-connection**：TCP リセットを送信し、TCP フローを乗っ取って終了します。**Reset TCP Connection** は、単一の接続を分析する TCP シグニチャでのみ機能します。スイープやフラッドに対しては機能しません。
- **modify-packet-inline**：パケット データを変更して、エンドポイントでパケットがどう処理されるかに関してあいまいな部分を除去します。

シグニチャのイベント アクションを設定するには、次の手順を実行します。

ステップ 1　管理者特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義モードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
sensor(config-sig)#
```

ステップ 3　設定するシグニチャを選択します。

```
sensor(config-sig)# signatures 1200 0
```

ステップ 4　Normalizer エンジンを入力します。

```
sensor(config-sig-sig)# engine normalizer
```

ステップ 5　イベント アクションを設定します。

```
sensor(config-sig-sig-nor)# event-action produce-alert|request-snmp-trap
```

（注）　シグニチャのイベント アクションを設定するたびに、前の設定が上書きされます。たとえば、シグニチャが反応したら必ずアラートを生成するには、その他の必要なイベント アクションとともにそのアクションを設定する必要があります。複数のイベント アクションを追加する場合は、| 記号を使用して、**product-alert|deny-packet-inline|request-snmp-trap** のようにします。

ステップ 6　設定を確認します。

```
sensor(config-sig-sig-nor)# show settings
   normalizer
   -----------------------------------------------
      event-action: produce-alert|request-snmp-trap default:
produce-alert|deny-packet-inline
```

ステップ 7　イベント アクション サブモードを終了します。

```
sensor(config-sig-sig-nor)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 8　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

AIC シグニチャの設定

この項では、AIC シグニチャとその設定方法について説明します。取り上げる事項は次のとおりです。

概要

AIC は Web トラフィックの詳細な分析を行います。HTTP プロトコルの不正利用を防止するために、HTTP セッションを精密に制御します。たとえば、インスタント メッセージや、gotomypc などのトンネリング アプリケーションなど、特定のポート上でトンネリングを行うアプリケーションに対する管理制御を行います。これらのアプリケーションが HTTP を介して稼働している場合は、P2P およびインスタント メッセージの検査とポリシー チェックを実行できます。

AIC は、FTP トラフィックを検査し、発行されるコマンドを制御する方法を提供します。

事前定義されたシグニチャをイネーブルまたはディセーブルにするか、カスタム シグニチャでポリシーを作成することができます。

（注）　AIC エンジンは、HTTP トラフィックが AIC Web ポートで受信されたときに実行されます。トラフィックが Web トラフィックであっても、AIC Web ポートで受信されない場合は、Service HTTP エンジンが実行されます。AIC 検査は、AIC Web ポートとして設定されている任意のポートで実行できます。検査されるトラフィックは HTTP トラフィックです。

AIC には、次のシグニチャのカテゴリがあります。

- HTTP 要求メソッド
  - 定義要求メソッド
  - 認識済み要求メソッド

  シグニチャの ID と説明については、P.7-18 の「AIC 要求メソッド シグニチャ」を参照してください。
- MIME タイプ
  - 定義コンテンツ タイプ
  - 認識されるコンテンツ タイプ

  シグニチャの ID と説明については、P.7-19 の「AIC MIME 定義コンテンツ タイプ シグニチャ」を参照してください。カスタム MIME シグニチャの作成手順については、P.7-43 の「AIC MIME タイプ シグニチャ」を参照してください。
- 定義 Web トラフィック ポリシー

  1 つの事前に定義されたシグニチャ 12674 があります。これは、非準拠の HTTP トラフィックが検出された場合に実行するアクションを指定しています。パラメータ Alarm on Non HTTP Traffic はシグニチャをイネーブルにします。デフォルトでは、このシグニチャはイネーブルです。
- 転送符号化
  - アクションと各メソッドを関連付けます。
  - センサーが認識済みのメソッドのリストを表示します。
  - チャンク符号化エラーが検出された場合に実行する必要のあるアクションを指定します。

  シグニチャの ID と説明については、P.7-22 の「AIC 転送符号化シグニチャ」を参照してください。
- FTP コマンド

  アクションを FTP コマンドに関連付けます。シグニチャの ID と説明については、P.7-22 の「AIC FTP コマンド シグニチャ」を参照してください。

アプリケーション ポリシーの設定

Web AIC 機能をイネーブルにするには、シグニチャ定義サブモードで **application-policy** コマンドを使用します。レイヤ 4 ～レイヤ 7 のパケット検査を実行するようにセンサーを設定して、Web および FTP サービスに関連した悪意のある攻撃を防ぐことができます。

次のオプションが適用されます。

- **ftp-enable [true | false]**：FTP サービスの保護をイネーブルにします。センサーで FTP トラフィックの検査を必須にするには、true に設定します。

  デフォルトは false です。
- **http-policy**：HTTP トラフィックの検査をイネーブルにします。
  - **aic-web-ports**：AIC トラフィックを探すポート用の変数。

    有効な範囲は、0 ～ 65535 です。a-b[,c-d] 形式で指定された、0 ～ 65535 の整数から成る範囲のカンマ区切りのリストです。範囲の 2 番目の数は、最初の数以上である必要があります。

    デフォルトは、80-80,3128-3128,8000-8000,8010-8010,8080-8080,8888-8888,24326-24326 です。
  - **http-enable [true | false]**：Web サービスに対応する保護をイネーブルにします。センサーで HTTP トラフィックが RFC に準拠しているかどうかの検査を必須にするには、true に設定します。

    デフォルトは false です。

- **max-outstanding-http-requests-per-connection** : 接続ごとに許可される HTTP 要求の最大数。有効な値は 1 ～ 16 です。デフォルトは 10 です。

アプリケーション ポリシーを設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　アプリケーション ポリシー サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
sensor(config-sig)# application-policy
```

ステップ 3　FTP トラフィックの検査をイネーブルにします。

```
sensor(config-sig-app)# ftp-enable true
```

ステップ 4　HTTP アプリケーション ポリシーを設定します。

a. HTTP アプリケーション ポリシー サブモードに入ります。

```
sensor(config-sig-app)# http-policy
```

b. HTTP アプリケーション ポリシーの実施をイネーブルにします。

```
sensor(config-sig-app-htt)# http-enable true
```

c. 各接続で、サーバからの応答を受信していない未完了の状態が許可される、未完了 HTTP 要求の数を指定します。

```
sensor(config-sig-app-htt)# max-outstanding-http-requests-per-connection 5
```

d. AIC ポートを編集します。

```
sensor(config-sig-app-htt)# aic-web-ports 80-80,3128-3128
```

ステップ 5　設定を確認します。

```
sensor(config-sig-app)# show settings
   application-policy
   ------------------------------------------------
      http-policy
      ------------------------------------------------
         http-enable: true default: false
         max-outstanding-http-requests-per-connection: 5 default: 10
         aic-web-ports: 80-80,3128-3128 default: 80-80,3128-3128,8000-8000,8010-
8010,8080-8080,8888-8888,24326-24326
      ------------------------------------------------
      ftp-enable: true default: false
   ------------------------------------------------
sensor(config-sig-app)#
```

ステップ 6　シグニチャ定義サブモードを終了します。

```
sensor(config-sig-app)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 7　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

AIC 要求メソッド シグニチャ

HTTP 要求メソッドには 2 つのシグニチャのカテゴリがあります。

- 定義要求メソッド：アクションと要求メソッドの関連付けを可能にします。シグニチャを拡張し変更できます（Define Request Method）。
- 認識済み要求メソッド：センサーが認識済みのメソッドのリストを表示します（Recognized Request Methods）。

表 7-1 に、事前定義済みの定義要求メソッド シグニチャを示します。必要とする事前定義済みのメソッドを持つシグニチャをイネーブルにしてください。シグニチャをイネーブルにする手順は、P.7-12 の「シグニチャのステータスの設定」を参照してください。

表 7-1　要求メソッド シグニチャ

シグニチャ ID	定義要求メソッド
12676	要求メソッドは認識されていない
12677	定義要求メソッド PUT
12678	定義要求メソッド CONNECT
12679	定義要求メソッド DELETE
12680	定義要求メソッド GET
12681	定義要求メソッド HEAD
12682	定義要求メソッド OPTIONS
12683	定義要求メソッド POST
12685	定義要求メソッド TRACE
12695	定義要求メソッド INDEX
12696	定義要求メソッド MOVE
12697	定義要求メソッド MKDIR
12698	定義要求メソッド COPY
12699	定義要求メソッド EDIT
12700	定義要求メソッド UNEDIT
12701	定義要求メソッド SAVE
12702	定義要求メソッド LOCK
12703	定義要求メソッド UNLOCK
12704	定義要求メソッド REVLABEL
12705	定義要求メソッド REVLOG
12706	定義要求メソッド REVADD
12707	定義要求メソッド REVNUM
12708	定義要求メソッド SETATTRIBUTE
12709	定義要求メソッド GETATTRIBUTENAME
12710	定義要求メソッド GETPROPERTIES
12711	定義要求メソッド STARTENV
12712	定義要求メソッド STOPREV

AIC MIME 定義コンテンツ タイプ シグニチャ

MIME タイプには 2 つのポリシーが関連付けられています。

- 定義コンテンツ タイプ：次の場合に特定のアクションを関連付けます（定義コンテンツ タイプ）。
  - image/jpeg などの特定の MIME タイプを拒否する
  - メッセージ サイズ違反
  - ヘッダーと本体に記述された MIME タイプが一致しない
- 認識されるコンテンツ タイプ（Recognized Content Type）

表 7-2 に、事前定義済みの定義コンテンツ タイプ シグニチャを示します。必要とする事前定義済みのメソッドを持つシグニチャをイネーブルにしてください。シグニチャをイネーブルにする手順は、P.7-12 の「シグニチャのステータスの設定」を参照してください。カスタム定義コンテンツ タイプ シグニチャを作成することもできます。手順については、P.7-43 の「AIC MIME タイプ シグニチャ」を参照してください。

表 7-2　定義コンテンツ タイプ シグニチャ

シグニチャ ID	シグニチャの説明
12621	コンテンツ タイプ image/gif のメッセージ長が無効です。
12622 2	コンテンツ タイプ image/png の検証に失敗しました。
12623 0 12623 1 12623 2	コンテンツ タイプ image/tiff のヘッダー チェック。 コンテンツ タイプ image/tiff のメッセージ長が無効です。 コンテンツ タイプ image/tiff の検証に失敗しました。
12624 0 12624 1 12624 2	コンテンツ タイプ image/x-3ds のヘッダー チェック。 コンテンツ タイプ image/x-3ds のメッセージ長が無効です。 コンテンツ タイプ image/x-3ds の検証に失敗しました。
12626 0 12626 1 12626 2	コンテンツ タイプ image/x-portable-bitmap のヘッダー チェック。 コンテンツ タイプ image/x-portable-bitmap のメッセージ長が無効です。 コンテンツ タイプ image/x-portable-bitmap の検証に失敗しました。
12627 0 12627 1 12627 2	コンテンツ タイプ image/x-portable-graymap のヘッダー チェック。 コンテンツ タイプ image/x-portable-graymap のメッセージ長が無効です。 コンテンツ タイプ image/x-portable-graymap の検証に失敗しました。
12628 0 12628 1 12628 2	コンテンツ タイプ image/jpeg のヘッダー チェック。 コンテンツ タイプ image/jpeg のメッセージ長が無効です。 コンテンツ タイプ image/jpeg の検証に失敗しました。
12629 0 12629 1	コンテンツ タイプ image/cgf のヘッダー チェック。 コンテンツ タイプ image/cgf のメッセージ長が無効です。
12631 0 12631 1	コンテンツ タイプ image/x-xpm のヘッダー チェック。 コンテンツ タイプ image/x-xpm のメッセージ長が無効です。
12633 0 12633 1 12633 2	コンテンツ タイプ audio/midi のヘッダー チェック。 無効なメッセージ長のコンテンツ タイプ audio/midi です。 コンテンツ タイプ audio/midi の検証に失敗しました。
12634 0 12634 1 12634 2	コンテンツ タイプ audio/basic のヘッダー チェック。 コンテンツ タイプ audio/basic のメッセージ長が無効です。 コンテンツ タイプ audio/basic の検証に失敗しました。
12635 0 12635 1 12635 2	コンテンツ タイプ audio/mpeg のヘッダー チェック。 コンテンツ タイプ audio/mpeg のメッセージ長が無効です。 コンテンツ タイプ audio/mpeg の検証に失敗しました。

表 7-2　定義コンテンツ タイプ シグニチャ（続き）

シグニチャ ID	シグニチャの説明
12636 0	コンテンツ タイプ audio/x-adpcm のヘッダー チェック。
12636 1	コンテンツ タイプ audio/x-adpcm のメッセージ長が無効です。
12636 2	コンテンツ タイプ audio/x-adpcm の検証に失敗しました。
12637 0	コンテンツ タイプ audio/x-aiff のヘッダー チェック。
12637 1	コンテンツ タイプ audio/x-aiff のメッセージ長が無効です。
12637 2	コンテンツ タイプ audio/x-aiff の検証に失敗しました。
12638 0	コンテンツ タイプ audio/x-ogg のヘッダー チェック。
12638 1	コンテンツ タイプ audio/x-ogg のメッセージ長が無効です。
12638 2	コンテンツ タイプ audio/x-ogg の検証に失敗しました。
12639 0	コンテンツ タイプ audio/x-wav のヘッダー チェック。
12639 1	コンテンツ タイプ audio/x-wav のメッセージ長が無効です。
12639 2	コンテンツ タイプ audio/x-wav の検証に失敗しました。
12641 0	コンテンツ タイプ text/html のヘッダー チェック。
12641 1	コンテンツ タイプ text/html のメッセージ長が無効です。
12641 2	コンテンツ タイプ text/html の検証に失敗しました。
12642 0	コンテンツ タイプ text/css のヘッダー チェック。
12642 1	コンテンツ タイプ text/css のメッセージ長が無効です。
12643 0	コンテンツ タイプ text/plain のヘッダー チェック。
12643 1	コンテンツ タイプ text/plain のメッセージ長が無効です。
12644 0	コンテンツ タイプ text/plain のヘッダー チェック。
12644 1	コンテンツ タイプ text/richtext のメッセージ長が無効です。
12645 0	コンテンツ タイプ text/sgml のヘッダー チェック。
12645 1	コンテンツ タイプ text/sgml のメッセージ長が無効です。
12645 2	コンテンツ タイプ text/sgml の検証に失敗しました。
12646 0	コンテンツ タイプ text/xml のヘッダー チェック。
12646 1	コンテンツ タイプ text/xml のメッセージ長が無効です。
12646 2	コンテンツ タイプ text/xml の検証に失敗しました。
12648 0	コンテンツ タイプ video/flc のヘッダー チェック。
12648 1	コンテンツ タイプ video/flc のメッセージ長が無効です。
12648 2	コンテンツ タイプ video/flc の検証に失敗しました。
12649 0	コンテンツ タイプ video/mpeg のヘッダー チェック。
12649 1	コンテンツ タイプ video/mpeg のメッセージ長が無効です。
12649 2	コンテンツ タイプ video/mpeg の検証に失敗しました。
12650 0	コンテンツ タイプ text/xmcd のヘッダー チェック。
12650 1	コンテンツ タイプ text/xmcd のメッセージ長が無効です。
12651 0	コンテンツ タイプ video/quicktime のヘッダー チェック。
12651 1	コンテンツ タイプ video/quicktime のメッセージ長が無効です。
12651 2	コンテンツ タイプ video/quicktime の検証に失敗しました。
12652 0	コンテンツ タイプ video/sgi のヘッダー チェック。
12652 1	コンテンツ タイプ video/sgi の検証に失敗しました。
12653 0	コンテンツ タイプ video/x-avi のヘッダー チェック。
12653 1	コンテンツ タイプ video/x-avi のメッセージ長が無効です。
12654 0	コンテンツ タイプ video/x-fli のヘッダー チェック。
12654 1	コンテンツ タイプ video/x-fli のメッセージ長が無効です。
12654 2	コンテンツ タイプ video/x-fli の検証に失敗しました。

表 7-2　定義コンテンツ タイプ シグニチャ（続き）

シグニチャ ID	シグニチャの説明
12655 0	コンテンツ タイプ video/x-mng のヘッダー チェック。
12655 1	コンテンツ タイプ video/x-mng のメッセージ長が無効です。
12655 2	コンテンツ タイプ video/x-mng の検証に失敗しました。
12656 0	コンテンツ タイプ application/x-msvideo のヘッダー チェック。
12656 1	コンテンツ タイプ application/x-msvideo のメッセージ長が無効です。
12656 2	コンテンツ タイプ application/x-msvideo の検証に失敗しました。
12658 0	コンテンツ タイプ application/ms-word のヘッダー チェック。
12658 1	コンテンツ タイプ application/ms-word のメッセージ長が無効です。
12659 0	コンテンツ タイプ application/octet-stream のヘッダー チェック。
12659 1	コンテンツ タイプ application/octet-stream のメッセージ長が無効です。
12660 0	コンテンツ タイプ application/postscript のヘッダー チェック。
12660 1	コンテンツ タイプ application/postscript のメッセージ長が無効です。
12660 2	コンテンツ タイプ application/postscript の検証に失敗しました。
12661 0	コンテンツ タイプ application/vnd.ms-excel のヘッダー チェック。
12661 1	コンテンツ タイプ application/vnd.ms-excel のメッセージ長が無効です。
12662 0	コンテンツ タイプ application/vnd.ms-powerpoint のヘッダー チェック。
12662 1	コンテンツ タイプ application/vnd.ms-powerpoint のメッセージ長が無効です。
12663 0	コンテンツ タイプ application/zip のヘッダー チェック。
12663 1	コンテンツ タイプ application/zip のメッセージ長が無効です。
12663 2	コンテンツ タイプ application/zip の検証に失敗しました。
12664 0	コンテンツ タイプ application/x-gzip のヘッダー チェック。
12664 1	コンテンツ タイプ application/x-gzip のメッセージ長が無効です。
12664 2	コンテンツ タイプ application/x-gzip の検証に失敗しました。
12665 0	コンテンツ タイプ application/x-java-archive のヘッダー チェック。
12665 1	コンテンツ タイプ application/x-java-archive のメッセージ長が無効です。
12666 0	コンテンツ タイプ application/x-java-vm のヘッダー チェック。
12666 1	コンテンツ タイプ application/x-java-vm のメッセージ長が無効です。
12667 0	コンテンツ タイプ application/pdf のヘッダー チェック。
12667 1	コンテンツ タイプ application/pdf のメッセージ長が無効です。
12667 2	コンテンツ タイプ application/pdf の検証に失敗しました。
12668 0	コンテンツ タイプ unknown のヘッダー チェック。
12668 1	コンテンツ タイプ unknown のメッセージ長が無効です。
12669 0	コンテンツ タイプ image/x-bitmap のヘッダー チェック。
12669 1	コンテンツ タイプ image/x-bitmap のメッセージ長が無効です。
12673 0	認識されるコンテンツ タイプ。

AIC 転送符号化シグニチャ

転送符号化に関連するポリシーは 3 つあります。

- アクションを各メソッドと関連付ける（Define Transfer Encoding）
- センサーによって認識されたメソッドをリストする（Recognized Transfer Encodings）
- チャンク符号化エラーが検出された場合に、どのアクションを実行するかを指定する（Chunked Transfer Encoding Error）

表 7-3 に、事前定義済みの転送符号化シグニチャを示します。必要な事前定義済み転送符号化メソッドがあるシグニチャをイネーブルにします。シグニチャをイネーブルにする手順は、P.7-12 の「シグニチャのステータスの設定」を参照してください。

表 7-3　転送符号化シグニチャ

シグニチャ ID	転送符号化メソッド
12686	Recognized Transfer Encoding
12687	Define Transfer Encoding Deflate
12688	Define Transfer Encoding Identity
12689	Define Transfer Encoding Compress
12690	Define Transfer Encoding GZIP
12693	Define Transfer Encoding Chunked
12694	Chunked Transfer Encoding Error

AIC FTP コマンド シグニチャ

表 7-4 に、事前定義済みの FTP コマンド シグニチャを示します。必要な事前定義 FTP コマンドを持つシグニチャをイネーブルにします。シグニチャをイネーブルにする手順は、P.7-12 の「シグニチャのステータスの設定」を参照してください。

表 7-4　FTP コマンド シグニチャ

シグニチャ ID	FTP コマンド
12900	認識されていない FTP コマンド
12901	FTP コマンド abor の定義
12902	FTP コマンド acct の定義
12903	FTP コマンド allo の定義
12904	FTP コマンド appe の定義
12905	FTP コマンド cdup の定義
12906	FTP コマンド cwd の定義
12907	FTP コマンド dele の定義
12908	FTP コマンド help の定義
12909	FTP コマンド list の定義
12910	FTP コマンド mkd の定義
12911	FTP コマンド mode の定義
12912	FTP コマンド nlst の定義
12913	FTP コマンド noop の定義
12914	FTP コマンド pass の定義
12915	FTP コマンド pasv の定義

表 7-4　FTP コマンド シグニチャ（続き）

シグニチャ ID	FTP コマンド
12916	FTP コマンド port の定義
12917	FTP コマンド pwd の定義
12918	FTP コマンド quit の定義
12919	FTP コマンド rein の定義
12920	FTP コマンド rest の定義
12921	FTP コマンド retr の定義
12922	FTP コマンド rmd の定義
12923	FTP コマンド rnfr の定義
12924	FTP コマンド rnto の定義
12925	FTP コマンド site の定義
12926	FTP コマンド smnt の定義
12927	FTP コマンド stat の定義
12928	FTP コマンド stor の定義
12929	FTP コマンド stou の定義
12930	FTP コマンド stru の定義
12931	FTP コマンド syst の定義
12932	FTP コマンド type の定義
12933	FTP コマンド user の定義

IP フラグメント再構成

この項では、IP フラグメント再構成について説明します。ここでは、IP フラグメント再構成シグニチャと設定可能なパラメータを示し、それらのパラメータの設定方法と IP フラグメント再構成の設定方法について説明します。取り上げる事項は次のとおりです。

概要

センサーは、複数のパケットにわたってフラグメント化されたデータグラムを再構成するように設定することができます。このとき、センサーが再構成するデータグラム フラグメントの数と、データグラムについてさらにフラグメントが届くのを待つ時間を判断するために使用する境界値を指定できます。これは、センサーがフレーム送信を受信できなかったため、または無作為にフラグメント化されたデータグラムを生成する攻撃が仕掛けられているために、完全に再組み立てができなくなっているデータグラムに、センサーのリソースをすべて割り当ててしまわないようにするためのものです。

IP フラグメント再構成はシグニチャごとに設定します。

IP フラグメント再構成シグニチャと設定可能なパラメータ

表 7-5 に、IP フラグメント再構成シグニチャと、このシグニチャに設定可能なパラメータを示します。IP フラグメント再構成シグニチャは、Normalizer エンジンの一部です。

表 7-5　IP フラグメント再構成シグニチャ

IP フラグメント再構成シグニチャ	デフォルト値のあるパラメータ
1200 IP Fragmentation Buffer Full	最大フラグメント 10000 を指定
1201 IP Fragment Overlap	なし
1202 IP Fragment Overrun - Datagram Too Long	最大データグラム サイズ 65536 を指定
1203 IP Fragment Overwrite - Data is Overwritten	なし
1204 IP Fragment Missing Initial Fragment	なし
1205 IP Fragment Too Many Datagrams	最大部分データグラム 1000 を指定
1206 IP Fragment Too Small	小さいフラグメントの最大値 2 を指定 最小フラグメント サイズ 400 を指定
1207 IP Fragment Too Many Datagrams	データグラムごとの最大フラグメント 170 を指定
1208 IP Fragment Incomplete Datagram	フラグメント再構成タイムアウト 60 を指定
1220 Jolt2 Fragment Reassembly DoS attack	最後のフラグメントの最大値 4 を指定
1225 Fragment Flags Invalid	なし

IP フラグメント再構成パラメータの設定

特定のシグニチャの IP フラグメント再構成パラメータを設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　IP フラグメント再構成シグニチャの ID とサブシグニチャ ID を指定します。

```
sensor(config-sig)# signatures 1200 0
```

ステップ 4　エンジンを指定します。

```
sensor(config-sig-sig)# engine normalizer
```

ステップ 5　編集デフォルト シグニチャ サブモードに入ります。

```
sensor(config-sig-sig-nor)# edit-default-sigs-only default-signatures-only
```

ステップ 6　任意の IP フラグメント再構成パラメータをイネーブルにし、必要に応じてデフォルト設定を変更します。たとえば、シグニチャ 1200 の最大フラグメント数を指定します。

```
sensor(config-sig-sig-nor-def)# specify-max-fragments yes
sensor(config-sig-sig-nor-def-yes)# max-fragments 20000
```

ステップ 7　設定を確認します。

```
sensor(config-sig-sig-nor-def-yes)# show settings
   yes
   -----------------------------------------------
      max-fragments: 20000 default: 10000
   -----------------------------------------------
sensor(config-sig-sig-nor-def-yes)#
```

ステップ 8　シグニチャ定義サブモードを終了します。

```
sensor(config-sig-sig-nor-def-yes)# exit
sensor(config-sig-sig-nor-def)# exit
sensor(config-sig-sig-nor)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 9　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

IP フラグメント再構成の方法の設定

センサーがフラグメントの再構成に使用する方法を設定するには、シグニチャ定義サブモードで **fragment-reassembly** コマンドを使用します。このオプションは、センサーが混合モードで動作しているときに設定できます。センサーがライン モードで動作している場合、このメソッドは NT 専用です。

次のオプションが適用されます。

- **ip-reassemble-mode**：センサーがフラグメントの再構成に使用する方法を、オペレーティング システムで指定します。
  - **nt**：Windows システム
  - **solaris**：Solaris システム
  - **linux**：GNU/Linux システム
  - **bsd**：BSD UNIX システム

    デフォルトは nt です。

IP フラグメント再構成を設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　フラグメント再構成サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
sensor(config-sig)# fragment-reassembly
```

ステップ 3　センサーで IP フラグメント再構成に使用するオペレーティング システムを設定します。

```
sensor(config-sig-fra)# ip-reassemble-mode linux
```

ステップ 4　設定を確認します。

```
sensor(config-sig-fra)# show settings
   fragment-reassembly
   -----------------------------------------------
      ip-reassemble-mode: linux default: nt
   -----------------------------------------------
sensor(config-sig-fra)#
```

ステップ 5　シグニチャ定義サブモードを終了します。

```
sensor(config-sig-fra)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 6　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

TCP ストリーム再構成の設定

この項では、TCP ストリーム再構成について説明し、設定可能なパラメータを持つ TCP ストリーム再構成シグニチャを示し、TCP ストリーム シグニチャの設定方法と TCP ストリーム再構成のモードの設定方法について説明します。取り上げる事項は次のとおりです。

概要

センサーは、完全な 3 ウェイ ハンドシェイクによって確立された TCP セッションだけを監視するように設定できます。また、ハンドシェークの完了まで待つ時間の最大値と、パケットがない場合に接続を監視し続ける時間も設定できます。これは、有効な TCP セッションが確立していないときにセンサーがアラートを生成しないようにするためのものです。センサーに対する攻撃には、単純に攻撃を繰り返すだけでセンサーにアラートを生成させようとするものがあります。TCP セッションの再組み立て機能は、センサーに対するこのような攻撃の緩和に役立ちます。

TCP ストリーム再構成パラメータは、シグニチャごとに設定します。TCP ストリーム再構成のモードを設定できます。

TCP ストリーム シグニチャと設定可能なパラメータ

表 7-6 に、TCP ストリーム再構成シグニチャと、TCP ストリーム再構成で設定可能なパラメータを示します。TCP ストリーム再構成シグニチャは、Normalizer エンジンの一部です。

表 7-6　TCP ストリーム再構成シグニチャ

TCP ストリーム再構成シグニチャ	デフォルト値のあるパラメータ
1300 TCP Segment Overwrite	なし
1301 TCP Session Inactivity Timeout	tcp-idle-timeout 3600
1302 TCP Session Embryonic Timeout	tcp-embryonic-timeout 15
1303 TCP Session Closing Timeout	tcp-closed-timeout 5
1304 TCP Session Packet Queue Overflow	tcp-max-queue 32
1305 TCP Urgent Flag Set	なし
1306 0 TCP Option Others 1306 1 TCP SACK Allowed Option 1306 2 TCP SACK Data Option 1306 3 TCP Timestamp Option 1306 4 TCP Window Scale Option 1306 5 TCP MSS Option	tcp-option-number 6-7,9-255
1307 TCP Window Size Variation	なし
1308 TTL Evasion	なし
1309 TCP Reserved Flags Set	なし
1310 TCP Retransmit Data Different	なし
1311 TCP Packet Exceeds MSS	なし
1312 TCP MSS Below Minimum	tcp-min-mss 400
1313 TCP MSS Exceed Maximum	tcp-max-mss 1460
1314 TCP SYN Packet with Data	なし
1330[1] 0 TCP Drop - Bad Checksum 1330 1 TCP Drop - Bad TCP Flags 1330 2 TCP Drop - Urgent Pointer Without Flag 1330 3 TCP Drop - Bad Option List 1330 4 TCP Drop - Bad Option Length 1330 5 TCP Drop - MSS Option in Non-SYN 1330 6 TCP Drop - WinScale Option in Non-SYN 1330 7 TCP Drop - Bad WinScale Option Value 1330 8 TCP Drop - Bad SACK Allow 1330 9 TCP Drop - Data in SYN|ACK 1330 10 TCP Drop - Data Past FIN 1330 11 TCP Drop - Timestamp not Allowed 1330 12 TCP Drop - Segment Out of Order 1330 13 TCP Drop - Invalid TCP Packet 1330 14 TCP Drop - RST or SYN in window 1330 15 TCP Drop - Segment Already ACKed by Peer 1330 16 TCP Drop - PAWS Check Failed 1330 17 TCP Drop - Segment out of State Order 1330 18 TCP Drop - Segment out of Window	なし
3050 Half Open SYN Attack	syn-flood-max-embryonic 5000

表 7-6　TCP ストリーム再構成シグニチャ（続き）

TCP ストリーム再構成シグニチャ	デフォルト値のあるパラメータ
3250 TCP Hijack	max-old-ack 200
3251 TCP Hijack Simplex Mode	max-old-ack 100

1. これらのサブシグニチャは、Normalizer エンジンが TCP パケットをドロップする理由を示しています。デフォルトでは、これらのサブシグニチャはパケットをドロップします。これらのサブシグニチャを使用すると、Normalizer エンジンで検査に合格しなかったパケットに IPS を通過させることができます。ドロップの理由は、TCP 統計情報内にエントリがあります。デフォルトでは、これらのサブシグニチャはアラートを生成しません。

TCP ストリーム再構成シグニチャの設定

特定のシグニチャに TCP ストリーム再構成を設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　TCP ストリーム再構成シグニチャの ID とサブシグニチャ ID を指定します。

```
sensor(config-sig)# signatures 1313 0
```

ステップ 4　エンジンを指定します。

```
sensor(config-sig-sig)# engine normalizer
```

ステップ 5　編集デフォルト シグニチャ サブモードに入ります。

```
sensor(config-sig-sig-nor)# edit-default-sigs-only default-signatures-only
```

ステップ 6　シグニチャ 1313 の最大 MSS パラメータをイネーブルにし、必要に応じてデフォルト設定を編集します。

```
sensor(config-sig-sig-nor-def)# specify-tcp-max-mss yes
sensor(config-sig-sig-nor-def-yes)# tcp-max-mss 1380
```

（注）　このパラメータをデフォルトの 1460 から 1380 へ変更すると、VPN トンネルを通過するトラフィックのフラグメント化を防ぐことができます。

ステップ 7　設定を確認します。

```
sensor(config-sig-sig-nor-def-yes)# show settings
   yes
   ----------------------------------------------
      tcp-max-mss: 1380 default: 1460
   ----------------------------------------------
sensor(config-sig-sig-nor-def-yes)#
```

ステップ 8　シグニチャ定義サブモードを終了します。

```
sensor(config-sig-sig-nor-def-yes)# exit
sensor(config-sig-sig-nor-def)# exit
sensor(config-sig-sig-nor)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 9　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

TCP ストリーム再構成のモードの設定

センサーが TCP セッションの再構成に使用するモードを設定するには、シグニチャ定義サブモードで **stream-reassembly** コマンドを使用します。

次のオプションが適用されます。

- **tcp-3-way-handshake-required [true | false]**：センサーが 3 ウェイ ハンドシェイクの完了したセッションだけを追跡するように指定します。
  デフォルトは true です。
- **tcp-reassembly-mode**：センサーが TCP セッションの再構成に使用するモードを指定します。
  - **strict**：シーケンスで次に予想されるものだけが許可されます。
  - **loose**：シーケンスに途切れがあっても許可されます。
  - **asym**：非同期トラフィックの再構成を許可します。
    デフォルトは strict です。

注意　非同期オプションを使用すると、TCP ウィンドウの回避チェックがディセーブルになります。

TCP ストリームの再構成パラメータを設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　TCP ストリーム再構成サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
sensor(config-sig)# stream-reassembly
```

ステップ 3　センサーが 3 ウェイ ハンドシェイクの完了したセッションだけを追跡するように指定します。

```
sensor(config-sig-str)# tcp-3-way-handshake-required true
```

ステップ 4　センサーが TCP セッションの再構成に使用するモードを指定します。

```
sensor(config-sig-str)# tcp-reassembly-mode strict
```

ステップ 5　設定を確認します。

```
sensor(config-sig-str)# show settings
   stream-reassembly
   -----------------------------------------------
      tcp-3-way-handshake-required: true default: true
      tcp-reassembly-mode: strict default: strict
   -----------------------------------------------
sensor(config-sig-str)#
```

ステップ 6　TCP ストリーム再構成サブモードを終了します。

```
sensor(config-sig-str)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 7　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

IP ロギングの設定

センサーが攻撃を検出したときに、IP セッション ログを生成するように設定できます。シグニチャの応答アクションとして IP ロギングが設定されているときにシグニチャが反応すると、アラートの送信元アドレスとの間で送受信されるすべてのパケットがログに記録されます。

IP ロギングを設定するには、シグニチャ定義サブモードで **ip-log** コマンドを使用します。

次のオプションが適用されます。

- **ip-log-bytes**：ログに記録する最大バイト数を指定します。
  有効な値は 0 ～ 2147483647 です。デフォルトは 0 です。
- **ip-log-packets**：ログに記録するパケット数を指定します。
  有効な値は 0 ～ 65535 です。デフォルトは 0 です。
- **ip-log-time**：センサーでログを記録する期間を指定します。
  有効な値は 30 ～ 300 秒です。デフォルトは 30 秒です。

（注）　センサーは、IP ロギング条件のいずれかを検出すると、IP ロギングを停止します。

IP ロギング パラメータを設定するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　IP ログ サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
sensor(config-sig)# ip-log
```

ステップ 3　IP ロギング パラメータを指定します。

a. ログに記録する最大バイト数を指定します。

```
sensor(config-sig-ip)# ip-log-bytes 200000
```

b. ログに記録するパケット数を指定します。

```
sensor(config-sig-ip)# ip-log-packets 150
```

c. センサーでログを記録する期間を指定します。

```
sensor(config-sig-ip)# ip-log-time 60
```

ステップ 4　設定を確認します。

```
sensor(config-sig-ip)# show settings
   ip-log
   -----------------------------------------------
      ip-log-packets: 150 default: 0
      ip-log-time: 60 default: 30
      ip-log-bytes: 200000 default: 0
   -----------------------------------------------
sensor(config-sig-ip)#
```

ステップ 5　IP ログ サブモードを終了します。

```
sensor(config-sig-ip)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 6　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

カスタム シグニチャの作成

この項では、カスタム シグニチャの作成方法について説明します。取り上げる事項は次のとおりです。

カスタム シグニチャの作成シーケンス

カスタム シグニチャを作成する場合は、次の手順に従います。

ステップ 1　シグニチャ エンジンを選択します。

ステップ 2　シグニチャの識別子を割り当てます。

- シグニチャ ID
- サブシグニチャ ID
- シグニチャ名
- アラートの注釈（オプション）
- ユーザ コメント（オプション）

ステップ 3　エンジン固有のパラメータを割り当てます。

パラメータはシグニチャ エンジンごとに異なりますが、各エンジンに適用されるマスター パラメータのグループが 1 つあります。

ステップ 4　アラート応答を割り当てます。

- シグニチャ忠実度評価
- アラートの重大度

ステップ 5　アラート動作を割り当てます。

ステップ 6　変更を適用します。

String TCP シグニチャの例

String エンジンは、ICMP、TCP、および UDP の各プロトコル用の汎用のパターン マッチング検査エンジンです。String エンジンは、複数のパターンを結合して単一のパターン マッチング テーブルにし、単一のデータ検索を可能にする、正規表現エンジンを使用します。

String ICMP、String TCP、および String UDP の 3 つの String エンジンがあります。

次の例は、カスタム String TCP シグニチャの作成方法を示しています。

（注）　この手順は、String UDP シグニチャおよび String ICMP シグニチャにも当てはまります。

String TCP エンジンには、次のパラメータが適用されます。

- **default**：値をシステム デフォルト設定に戻します。
- **direction**：トラフィックの方向。
  - **from-service**：サービス ポートからクライアント ポートに向かうトラフィック。
  - **to-service**：クライアント ポートからサービス ポートに向かうトラフィック。
- **event-action**：アラートがトリガーされたときに実行するアクション。
  - **deny-attacker-inline**：（インライン モードのみ）指定された期間、攻撃者アドレスから、このパケットおよび将来のパケットを送信しません。
  - **deny-attacker-service-pair-inline**：（インラインのみ）指定された期間、攻撃者アドレスと被害先のポートのペアで、このパケットおよび将来のパケットを送信しません。
  - **deny-attacker-victim-pair-inline**：（インラインのみ）指定された期間、攻撃者 / 被害先のアドレスのペアで、このパケットおよび将来のパケットを送信しません。
  - **deny-connection-inline**：（インライン モードのみ）TCP フローで、このパケットおよび将来のパケットを送信しません。
  - **deny-packet-inline**：（インライン モードのみ）このパケットを送信しません。
  - **log-attacker-packets**：攻撃者のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **log-pair-packets**：攻撃者と被害先のアドレスのペアを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **log-victim-packets**：被害先のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **produce-alert**：イベントをアラートとしてイベント ストアに書き込みます。
  - **produce-verbose-alert**：違反パケットの符号化ダンプ（切り捨てられている可能性があります）をアラートに組み込みます。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **request-block-connection**：この接続をブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-block-host**：この攻撃者ホストをブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-rate-limit**：レート制限を実行するレート制限要求を ARC に送信します。レート制限デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-snmp-trap**：SNMP 通知を実行する要求をセンサーの通知アプリケーション コマンドに送信します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。SNMP は、このアクションを実装するようセンサーで設定されている必要があります。
  - **reset-tcp-connection**：TCP リセットを送信し、TCP フローを乗っ取って終了します。**Reset TCP Connection** は、単一の接続を分析する TCP シグニチャでのみ機能します。スイープやフラッドに対しては機能しません。
  - **modify-packet-inline**：パケット データを変更して、エンドポイントでパケットがどう処理されるかに関してあいまいな部分を除去します。
- **no**：エントリまたは選択設定を削除します。
- **regex-string**：単一の TCP パケット内で検索する正規表現。

- **service-ports**：ターゲット サービスが常駐できるポートまたはポート範囲。

  有効な範囲は、0 ～ 65535 です。a-b[,c-d] 形式で指定された、0 ～ 65535 の整数から成る範囲のカンマ区切りのリストです。範囲の 2 番目の数は、最初の数以上である必要があります。
- **specify-exact-match-offset [yes | no]**：（オプション）完全一致オフセットをイネーブルにします。
- **specify-min-match-length [yes | no]**：（オプション）最小一致長をイネーブルにします。
- **strip-telnet-options**：検索の前にデータから Telnet オプション文字を削除します。
- **swap-attacker-victim [true | false]**：アラーム メッセージでアドレス（およびポート）ソースと宛先をスワップするかどうかを示します。デフォルトは、スワッピングなしを意味する false です。

String TCP エンジンに基づいてシグニチャを作成するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　シグニチャのシグニチャ ID とサブシグニチャ ID を指定します。

```
sensor(config-sig)# signatures 60025 0
```

カスタム シグニチャの範囲は 60000 ～ 65000 です。

ステップ 4　シグニチャ説明サブモードに入ります。

```
sensor(config-sig-sig)# sig-description
```

ステップ 5　新しいシグニチャの名前を指定します。

```
sensor(config-sig-sig-sig)# sig-name This is my new name
```

また、**sig-comment** コマンドを使用してシグニチャの追加コメントを指定したり、**sig-string-info** コマンドを使用してシグニチャに関する追加情報を指定したりすることもできます。

ステップ 6　シグニチャ説明サブモードを終了します。

```
sensor(config-sig-sig-sig)# exit
```

ステップ 7　String TCP エンジンを指定します。

```
sensor(config-sig-sig)# engine string-tcp
```

ステップ 8　サービス ポートを指定します。

```
sensor(config-sig-sig-str)# service-ports 23
```

ステップ 9　方向を指定します。

```
sensor(config-sig-sig-str)# direction to-service
```

ステップ 10　TCP パケット内で検索する正規表現文字列を指定します。

```
sensor(config-sig-sig-str)# regex-string This-is-my-new-Sig-regex
```

ステップ 11　必要れあれば、**event-action** コマンドを使用して、セキュリティ ポリシーに従ってイベント アクションを変更することができます。デフォルトのイベント アクションは **produce-alert** です。

ステップ 12　このカスタム String TCP シグニチャでは、次のオプション パラメータを変更することができます。

- **specify-exact-match-offset**
- **specify-min-match-length**
- **strip-telnet-options**
- **swap-attacker-victim**

ステップ 13　設定を確認します。

```
sensor(config-sig-sig-str)# show settings
   string-tcp
   -----------------------------------------------
      event-action: produce-alert <defaulted>
      strip-telnet-options: false <defaulted>
      specify-min-match-length
      -----------------------------------------------
         no
         ---------------------------------------------
         ---------------------------------------------
      -----------------------------------------------
      regex-string: This-is-my-new-Sig-regex
      service-ports: 23
      direction: to-service default: to-service
      specify-exact-match-offset
      -----------------------------------------------
         no
         ---------------------------------------------
            specify-max-match-offset
            ------------------------------------------
               no
               -----------------------------------------
               -----------------------------------------
            ------------------------------------------
            specify-min-match-offset
            ------------------------------------------
               no
               -----------------------------------------
               -----------------------------------------
            ------------------------------------------
         ---------------------------------------------
      -----------------------------------------------
      swap-attacker-victim: false <defaulted>
   -----------------------------------------------
sensor(config-sig-sig-str)#
```

ステップ 14 シグニチャ定義サブモードを終了します。

```
sensor(config-sig-sig-str)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 15 変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

Service HTTP シグニチャの例

Service HTTP エンジンは、サービス固有で文字列ベースのパターン マッチング検査エンジンです。HTTP プロトコルは、今日のネットワークで最もよく使用されているプロトコルです。また、これには最も長い事前処理時間が必要であり、検査を必要とする最大数のシグニチャを持つため、システムのパフォーマンス全体にとって重大となります。

Service HTTP エンジンは、複数のパターンを結合して単一のパターン マッチング テーブルにし、単一のデータ検索を可能にする、正規表現ライブラリを使用します。このエンジンは、Web サービスだけに送られるトラフィック、または HTTP 要求を検索します。このエンジンでリターン トラフィックを検査することはできません。このエンジンの各シグニチャで、該当する個別の Web ポートを指定できます。

HTTP 解読は、符号化された文字を ASCII 対応文字に正規化することによって、HTTP メッセージをデコードするプロセスです。これは、ASCII 正規化とも呼ばれます。

HTTP パケットを検査するには、まずデータを、ターゲット システムでデータの処理時に表示されるものと同じ表記に解読または正規化する必要があります。どのオペレーティング システムおよび Web サーバ バージョンがターゲットで動作しているかを認識している、カスタマイズされたデコード技術をホスト ターゲット タイプごとに用意することをお勧めします。Service HTTP エンジンには、Microsoft IIS Web サーバに対するデフォルトの解読動作があります。

次の例は、カスタム Service HTTP シグニチャの作成方法を示しています。

Service HTTP エンジンには、次のオプションが適用されます。

- **de-obfuscate [true | false]**：検索の前に反回避解読を適用します。
- **default**：値をシステム デフォルト設定に戻します。
- **event-action**：アラートがトリガーされたときに実行するアクション。
  - **deny-attacker-inline**：（インライン モードのみ）指定された期間、攻撃者アドレスから、このパケットおよび将来のパケットを送信しません。
  - **deny-attacker-service-pair-inline**：（インラインのみ）指定された期間、攻撃者アドレスと被害先のポートのペアで、このパケットおよび将来のパケットを送信しません。
  - **deny-attacker-victim-pair-inline**：（インラインのみ）指定された期間、攻撃者 / 被害先のアドレスのペアで、このパケットおよび将来のパケットを送信しません。
  - **deny-connection-inline**：（インライン モードのみ）TCP フローで、このパケットおよび将来のパケットを送信しません。
  - **deny-packet-inline**：（インライン モードのみ）このパケットを送信しません。
  - **log-attacker-packets**：攻撃者のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。

- **log-pair-packets** : 攻撃者と被害先のアドレスのペアを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
- **log-victim-packets** : 被害先のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
- **produce-alert** : イベントをアラートとしてイベント ストアに書き込みます。
- **produce-verbose-alert** : 違反パケットの符号化ダンプ（切り捨てられている可能性があります）をアラートに組み込みます。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
- **request-block-connection** : この接続をブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
- **request-block-host** : この攻撃者ホストをブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
- **request-rate-limit** : レート制限を実行するレート制限要求を ARC に送信します。レート制限デバイスは、このアクションを実装するよう設定されている必要があります。
- **request-snmp-trap** : SNMP 通知を実行する要求をセンサーの通知アプリケーション コマンドに送信します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。SNMP は、このアクションを実装するようセンサーで設定されている必要があります。
- **reset-tcp-connection** : TCP リセットを送信し、TCP フローを乗っ取って終了します。**Reset TCP Connection** は、単一の接続を分析する TCP シグニチャでのみ機能します。スイープやフラッドに対しては機能しません。
- **modify-packet-inline** : パケット データを変更して、エンドポイントでパケットがどう処理されるかに関してあいまいな部分を除去します。

- **max-field-sizes** : 最大フィールド サイズのグループ。
  - **specify-max-arg-field-length [yes | no]** : max-arg-field-length をイネーブルにします（オプション）。
  - **specify-max-header-field-length [yes | no]** : max-header-field-length をイネーブルにします（オプション）。
  - **specify-max-request-length [yes | no]** : max-request-length をイネーブルにします（オプション）。
  - **specify-max-uri-field-length [yes | no]** : max-uri-field-length をイネーブルにします（オプション）。
- **no** : エントリまたは選択設定を削除します。
- **regex** : 正規表現のグループ。
  - **specify-arg-name-regex** : arg-name-regex をイネーブルにします（オプション）。
  - **specify-header-regex** : header-regex をイネーブルにします（オプション）。
  - **specify-request-regex** : request-regex をイネーブルにします（オプション）。
  - **specify-uri-regex** : uri-regex をイネーブルにします（オプション）。
- **service-ports** : ターゲット サービスが常駐することのできるポートまたはポート範囲のカンマ区切りリスト。
- **swap-attacker-victim [true | false]** : アラーム メッセージでアドレス（およびポート）ソースと宛先をスワップするかどうかを示します。デフォルトは、スワッピングなしを意味する false です。

Service HTTP エンジンに基づいてカスタム シグニチャを作成するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　シグニチャのシグニチャ ID とサブシグニチャ ID を指定します。

```
sensor(config-sig)# signatures 63000 0
```

カスタム シグニチャの範囲は 60000 ～ 65000 です。

ステップ 4　シグニチャ説明モードに入ります。

```
sensor(config-sig-sig)# sig-description
```

ステップ 5　シグニチャ名を指定します。

```
sensor(config-sig-sig-sig)# sig-name myWebSig
```

ステップ 6　アラートの特性を指定します。

```
sensor(config-sig-sig-sig)# alert-traits 2
```

有効な範囲は 0 ～ 65535 です。

ステップ 7　シグニチャ説明サブモードを終了します。

```
sensor(config-sig-sig-sig)# exit
```

ステップ 8　アラート頻度を割り当てます。

```
sensor(config-sig-sig)# alert-frequency
sensor(config-sig-sig-ale)# summary-mode fire-all
sensor(config-sig-sig-ale-fir)# summary-key Axxx
sensor(config-sig-sig-ale-fir)# specify-summary-threshold yes
sensor(config-sig-sig-ale-fir-yes)# summary-threshold 200
```

ステップ 9　アラート頻度サブモードを終了します。

```
sensor(config-sig-sig-ale-fir-yes)# exit
sensor(config-sig-sig-ale-fir)# exit
sensor(config-sig-sig-ale)# exit
```

ステップ 10　検索の前に反回避解読を適用するようにシグニチャを設定します。

```
sensor(config-sig-sig)# engine service-http
sensor(config-sig-sig-ser)# de-obfuscate true
```

ステップ 11 正規表現パラメータを設定します。

```
sensor(config-sig-sig)# engine service-http
sensor(config-sig-sig-ser)# regex
sensor(config-sig-sig-ser-reg)# specify-uri-regex yes
sensor(config-sig-sig-ser-reg-yes)# uri-regex [Mm][Yy][Ff][Oo][Oo]
```

ステップ 12 正規表現サブモードを終了します。

```
sensor(config-sig-sig-ser-reg-yes)# exit
sensor(config-sig-sig-ser-reg-)# exit
```

ステップ 13 シグニチャ変数 WEBPORTS を使用するサービス ポートを設定します。

```
sensor(config-sig-sig-ser)# service-ports $WEBPORTS
```

ステップ 14 シグニチャ定義サブモードを終了します。

```
sensor(config-sig-sig-ser)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 15 変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

MEG シグニチャの例

Meta エンジンは、スライドする時間間隔内で、関連する方法で発生するイベントを定義します。このエンジンは、パケットではなくイベントを処理します。シグニチャ イベントが生成されると、Meta エンジンがそれらを検査して、1 つまたはいくつかの Meta 定義と一致するかどうかを判別します。Meta エンジンは、このイベントに対するすべての要件が満たされた後に、シグニチャ イベントを生成します。

シグニチャ イベントはすべて SEAP によって Meta エンジンに渡されます。SEAP は、minimum hits オプションを処理した後で、イベントを渡します。要約とイベント アクションは、Meta エンジンがコンポーネント イベントを処理した後に処理されます。SEAP の詳細については、P.6-2 の「Signature Event Action Processor」を参照してください。

注意　多数の Meta シグニチャが、意図せずセンサー パフォーマンス全体に影響を及ぼす可能性があります。

次の例は、Meta エンジンに基づいて MEG シグニチャを作成する方法を示しています。

（注）　Meta エンジンは、ほとんどのエンジンが入力としてパケットを取るのに対し、入力としてアラートを取る点で、その他のエンジンとは異なります。Meta エンジンの詳細については、P.B-14 の「Meta エンジン」を参照してください。

Meta シグニチャ エンジンには、次のオプションが適用されます。

- **component-list**：Meta コンポーネントのリスト。
  - **edit**：リスト内の既存のエントリを編集します。
  - **insert** *name1*：リストに新しいエントリを挿入します。
  - **move**：リスト内のエントリを移動します。
  - **begin**：エントリをアクティブ リストの先頭に配置します。
  - **end**：エントリをアクティブ リストの終わりに配置します。
  - **inactive**：エントリを非アクティブ リストに配置します。
  - **before**：エントリを指定したエントリの前に配置します。
  - **after**：エントリを指定したエントリの後ろに配置します。
- **component-count**：このコンポーネントが満たされるまでにコンポーネントが反応する回数。
- **component-sig-id**：このコンポーネントを照合するシグニチャのシグニチャ ID。
- **component-subsig-id**：このコンポーネントを照合するシグニチャのサブシグニチャ ID。
- **component-list-in-order [true | false]**：コンポーネント リストの順に反応するかどうかの指定。
- **event-action**：アラートがトリガーされたときに実行するアクション。
  - **deny-attacker-inline**：（インライン モードのみ）指定された期間、攻撃者アドレスから、このパケットおよび将来のパケットを送信しません。
  - **deny-attacker-service-pair-inline**：（インラインのみ）指定された期間、攻撃者アドレスと被害先のポートのペアで、このパケットおよび将来のパケットを送信しません。
  - **deny-attacker-victim-pair-inline**：（インラインのみ）指定された期間、攻撃者 / 被害先のアドレスのペアで、このパケットおよび将来のパケットを送信しません。
  - **deny-connection-inline**：（インライン モードのみ）TCP フローで、このパケットおよび将来のパケットを送信しません。
  - **deny-packet-inline**：（インライン モードのみ）このパケットを送信しません。
  - **log-attacker-packets**：攻撃者のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **log-pair-packets**：攻撃者と被害先のアドレスのペアを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **log-victim-packets**：被害先のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **produce-alert**：イベントをアラートとしてイベント ストアに書き込みます。
  - **produce-verbose-alert**：違反パケットの符号化ダンプ（切り捨てられている可能性があります）をアラートに組み込みます。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **request-block-connection**：この接続をブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-block-host**：この攻撃者ホストをブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-rate-limit**：レート制限を実行するレート制限要求を ARC に送信します。レート制限デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-snmp-trap**：SNMP 通知を実行する要求をセンサーの通知アプリケーション コマンドに送信します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。SNMP は、このアクションを実装するようセンサーで設定されている必要があります。

  - **reset-tcp-connection**：TCP リセットを送信し、TCP フローを乗っ取って終了します。**Reset TCP Connection** は、単一の接続を分析する TCP シグニチャでのみ機能します。スイープやフラッドに対しては機能しません。
  - **modify-packet-inline**：パケット データを変更して、エンドポイントでパケットがどう処理されるかに関してあいまいな部分を除去します。
- **meta-key**：Meta シグニチャのストレージ タイプ。
  - **AaBb**：攻撃者と被害先のアドレスとポート。
  - **AxBx**：攻撃者と被害先のアドレス。
  - **Axxx**：攻撃者のアドレス。
  - **xxBx**：被害先のアドレス。
- **meta-reset-interval**：Meta シグニチャをリセットする間隔（秒単位）。
  有効な範囲は 0 ～ 3600 秒です。デフォルトは 60 秒です。

（注） シグニチャ 64000 のサブシグニチャ 0 は、同じ送信元アドレスのシグニチャ 2000 のサブシグニチャ 0 およびシグニチャ 3000 のサブシグニチャ 0 からのアラートを認識すると反応します。送信元アドレス選択は、メタ キーのデフォルト値 Axxx の結果です。メタ キー設定を xxBx（宛先アドレス）に変更することによって、動作を変更できます。たとえば、次のようになります。

Meta エンジンに基づいて MEG シグニチャを作成するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　シグニチャ定義サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
```

ステップ 3　シグニチャのシグニチャ ID とサブシグニチャ ID を指定します。

```
sensor(config-sig)# signatures 64000 0
```

カスタム シグニチャの範囲は 60000 ～ 65000 です。

ステップ 4　シグニチャ エンジンを指定します。

```
sensor(config-sig-sig)# engine meta
```

ステップ 5　リストの先頭に MEG シグニチャ（名前は c1）を挿入します。

```
sensor(config-sig-sig-met)# component-list insert c1 begin
```

ステップ 6　このコンポーネントを照合するシグニチャのシグニチャ ID を指定します。

```
sensor(config-sig-sig-met-com)# component-sig-id 2000
```

ステップ 7　コンポーネント リスト サブモードを終了します。

```
sensor(config-sig-sig-met-com)# exit
```

ステップ 8　別の MEG シグニチャ（名前は c2）をリストの最後に挿入します。

```
sensor(config-sig-sig-met)# component-list insert c2 end
```

ステップ 9　このコンポーネントを照合するシグニチャのシグニチャ ID を指定します。

```
sensor(config-sig-sig-met-com)# component-sig-id 3000
```

ステップ 10　設定を確認します。

```
sensor(config-sig-sig-met-com)# exit
sensor(config-sig-sig-met)# show settings
   meta
   -----------------------------------------------
      event-action: produce-alert <defaulted>
      meta-reset-interval: 60 <defaulted>
      component-list (min: 1, max: 8, current: 2 - 2 active, 0 inactive)
      -----------------------------------------------
      ACTIVE list-contents
      -----------------------------------------------
         NAME: c1
         -----------------------------------------------
            component-sig-id: 2000
            component-subsig-id: 0 <defaulted>
            component-count: 1 <defaulted>
         -----------------------------------------------
         -----------------------------------------------
         NAME: c2
         -----------------------------------------------
            component-sig-id: 3000
            component-subsig-id: 0 <defaulted>
            component-count: 1 <defaulted>
         -----------------------------------------------
         -----------------------------------------------
      -----------------------------------------------
      meta-key
      -----------------------------------------------
         Axxx
         -----------------------------------------------
            unique-victims: 1 <defaulted>
         -----------------------------------------------
      -----------------------------------------------
      component-list-in-order: false <defaulted>
   -----------------------------------------------
sensor(config-sig-sig-met)#
```

ステップ 11　シグニチャ定義サブモードを終了します。

```
sensor(config-sig-sig-met)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 12　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。

AIC MIME タイプ シグニチャ

次の例は、AIC エンジンに基づいて MIME タイプ シグニチャを作成する方法を示しています。

次のオプションが適用されます。

- **event-action**：アラートがトリガーされたときに実行するアクション。
  - **deny-attacker-inline**：（インライン モードのみ）指定された期間、攻撃者アドレスから、このパケットおよび将来のパケットを送信しません。
  - **deny-attacker-service-pair-inline**：（インラインのみ）指定された期間、攻撃者アドレスと被害先のポートのペアで、このパケットおよび将来のパケットを送信しません。
  - **deny-attacker-victim-pair-inline**：（インラインのみ）指定された期間、攻撃者 / 被害先のアドレスのペアで、このパケットおよび将来のパケットを送信しません。
  - **deny-connection-inline**：（インライン モードのみ）TCP フローで、このパケットおよび将来のパケットを送信しません。
  - **deny-packet-inline**：（インライン モードのみ）このパケットを送信しません。
  - **log-attacker-packets**：攻撃者のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **log-pair-packets**：攻撃者と被害先のアドレスのペアを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **log-victim-packets**：被害先のアドレスを含むパケットの IP ロギングを開始します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **produce-alert**：イベントをアラートとしてイベント ストアに書き込みます。
  - **produce-verbose-alert**：違反パケットの符号化ダンプ（切り捨てられている可能性があります）をアラートに組み込みます。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。
  - **request-block-connection**：この接続をブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-block-host**：この攻撃者ホストをブロックする要求を ARC に送信します。ブロッキング デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-rate-limit**：レート制限を実行するレート制限要求を ARC に送信します。レート制限デバイスは、このアクションを実装するよう設定されている必要があります。
  - **request-snmp-trap**：SNMP 通知を実行する要求をセンサーの通知アプリケーション コマンドに送信します。このアクションを実行すると、**produce-alert** が選択されていない場合でも、イベント ストアにアラートが書き込まれます。SNMP は、このアクションを実装するようセンサーで設定されている必要があります。
  - **reset-tcp-connection**：TCP リセットを送信し、TCP フローを乗っ取って終了します。**Reset TCP Connection** は、単一の接続を分析する TCP シグニチャでのみ機能します。スイープやフラッドに対しては機能しません。
  - **modify-packet-inline**：パケット データを変更して、エンドポイントでパケットがどう処理されるかに関してあいまいな部分を除去します。
- **no**：エントリまたは選択設定を削除します。
- **signature-type**：目的のシグニチャ タイプ。
  - **content-types**：コンテンツ タイプ。
  - **define-web-traffic-policy**：Web トラフィック ポリシーの定義。
  - **max-outstanding-requests-overrun**：多数の未完了 HTTP 要求の検査。
  - **msg-body-pattern**：メッセージ本体のパターン。
  - **request-methods**：要求メソッドを処理するシグニチャ タイプ。

－ **transfer-encodings**：転送符号化を処理するシグニチャ タイプ。

MIME タイプ ポリシーのシグニチャを定義するには、次の手順を実行します。

ステップ 1　管理者特権またはオペレータ特権を持つアカウントを使用して CLI にログインします。

ステップ 2　アプリケーション ポリシー実施サブモードに入ります。

```
sensor# configure terminal
sensor(config)# service signature-definition sig0
sensor(config-sig)# signatures 60001 0
sensor(config-sig-sig)# engine application-policy-enforcement-http
```

ステップ 3　イベント アクションを指定します。

```
sensor(config-sig-sig-app)# event-action produce-alert|log-pair-packets
```

ステップ 4　シグニチャ タイプを定義します。

```
sensor(config-sig-sig-app)# signature-type content-type define-content-type
```

ステップ 5　コンテンツ タイプを定義します。

```
sensor(config-sig-sig-app-def)# name MyContent
```

ステップ 6　設定を確認します。

```
sensor(config-sig-sig-app-def)# show settings
-> define-content-type
   -----------------------------------------------
      name: MyContent
*---> content-type-details
      -----------------------------------------------
      -----------------------------------------------
   -----------------------------------------------
sensor(config-sig-sig-app-def)#
```

ステップ 7　シグニチャ サブモードを終了します。

```
sensor(config-sig-sig-app-def)# exit
sensor(config-sig-sig-app)# exit
sensor(config-sig-sig)# exit
sensor(config-sig)# exit
Apply Changes:?[yes]:
```

ステップ 8　変更を適用する場合は **Enter** キーを押し、変更を廃棄する場合は **no** と入力します。