CHAPTER 20

フィルタリング サービスの適用

この章では、Web トラフィックをフィルタリングして、セキュリティ リスクを低減し、不適切な使用を回避する方法について説明します。この章の内容は、次のとおりです。

フィルタリングの概要

この項では、フィルタリングを適用することにより、セキュリティ アプライアンスを通過するトラフィックをどのように制御できるかについて説明します。フィルタリングは、次の 2 つの異なる方法で使用できます。

- ActiveX オブジェクトまたは Java アプレットのフィルタリング
- 外部フィルタリング サーバを使用するフィルタリング

アクセスを全面的にブロックする代わりに、ActiveX オブジェクトや Java アプレットなど、特定の状況でセキュリティ上の脅威をもたらす可能性のある特定の不適切なオブジェクトを HTTP トラフィックから取り除くことができます。

URL フィルタリングを使用して、Secure Computing SmartFilter（従来の N2H2）や Websense などの外部フィルタリング サーバに特定のトラフィックを誘導することもできます。長い URL、HTTPS および FTP フィルタリングは、Websense および Secure Computing SmartFilter の両方を使って、URL フィルタリングのためにイネーブルにできるようになりました。フィルタリング サーバは、セキュリティ ポリシーで指定されている特定のサイトまたは特定のタイプのサイトに向かうトラフィックをブロックできます。

（注） URL キャッシングが動作するのは、URL サーバのベンダーから提供された URL サーバ ソフトウェアのバージョンで URL キャッシングがサポートされている場合だけです。

URL フィルタリングは CPU に大きな負荷がかかるため、外部フィルタリング サーバを使用することにより、他のトラフィックのスループットに影響を与えることがなくなります。ただし、外部フィルタリング サーバを使用してトラフィックをフィルタリングしている場合でも、ネットワークの速度および URL フィルタリング サーバのキャパシティによっては、最初の接続に必要な時間が著しく長くなる場合もあります。

ActiveX オブジェクトのフィルタリング

この項では、フィルタリングを適用して、ファイアウォールを通過する HTTP トラフィックから ActiveX オブジェクトを取り除く方法について説明します。この項では、次のトピックについて取り上げます。

ActiveX のフィルタリングの概要

ActiveX オブジェクトには、保護されているネットワーク上のホストやサーバを攻撃することを目的とするコードが含まれている場合があるため、セキュリティのリスクが発生する可能性があります。ActiveX オブジェクトは、ActiveX フィルタリングでディセーブルにできます。

ActiveX コントロールは、以前は OLE コントロールまたは OCX コントロールと呼ばれていたもので、Web ページやその他のアプリケーションに挿入できるコンポーネントです。これらのコントロールにはカスタム フォームやカレンダーなど、情報の収集と表示に使用されるサードパーティ製の多様なフォームが含まれています。ActiveX は、技術的に、ネットワーク クライアントに対して多くの問題を発生させる可能性があります。たとえば、ワークステーションの障害の原因となる、ネットワーク セキュリティ問題を引き起こす、またはサーバへの攻撃に利用される、などのおそれがあります。

filter activex コマンドは、HTML <object> コマンドを、HTML Web ページ内でコメントアウトすることでブロックします。<APPLET> ～ </APPLET> タグおよび <OBJECT CLASSID> ～ </OBJECT> タグを選択的にコメントに置換することによって、HTML ファイルの ActiveX フィルタリングが実行されます。ネストされたタグのフィルタリングは、最上位タグをコメントに変換することによってサポートされています。

注意　このコマンドは、オブジェクト タグに埋め込まれている Java アプレット、イメージ ファイル、またはマルチメディア オブジェクトもすべてブロックします。

<object> または </object> という HTML タグが複数のネットワーク パケットに分割されている場合、またはタグ内のコードが MTU のバイト数より長い場合、セキュリティ アプライアンスはそのタグをブロックできません。

alias コマンドによって参照されている IP アドレスにユーザがアクセスした場合、または WebVPN トラフィックでは、ActiveX ブロッキングは行われません。

ActiveX フィルタリングのイネーブル化

ここでは、セキュリティ アプライアンス を通過する HTTP トラフィック内の ActiveX オブジェクトを削除する方法について説明します。ActiveX オブジェクトを削除するには、グローバル コンフィギュレーション モードで次のコマンドを入力します。

```
hostname(config)# filter activex port[-port] local_ip local_mask foreign_ip foreign_mask
```

このコマンドを使用するには、*port* に、フィルタリングを適用する TCP ポートを指定します。一般的に、これはポート 80 ですが、他の値でも受け入れられます。ポート 80 には **http** または **url** リテラルを使用できます。ポート範囲を指定するには、開始ポート番号と終了ポート番号の間にハイフンを使用します。

ローカル IP アドレスおよびマスクによって、フィルタリングされるトラフィックの発信元である 1 台以上の内部ホストを指定します。外部アドレスおよびマスクは、フィルタリングされるトラフィックの外部の宛先を指定します。

これらのアドレスに **0.0.0.0**（短縮形は **0**）を設定して、すべてのホストを指定できます。これらのマスクに **0.0.0.0**（短縮形は **0**）を使用して、すべてのホストを指定できます。

次に、すべての発信接続で ActiveX オブジェクトをブロックする例を示します。

```
hostname(config)# filter activex 80 0 0 0 0
```

このコマンドは、任意のローカル ホストから任意の外部ホストへの接続において、ポート 80 で Web トラフィックに対して ActiveX オブジェクト ブロッキングを適用することを指定します。

このコンフィギュレーションを削除するには、次の例で示すように、コマンドの **no** 形式を使用します。

```
hostname(config)# no filter activex 80 0 0 0 0
```

Java アプレットのフィルタリング

ここでは、フィルタリングを適用して、ファイアウォールを通過する HTTP トラフィックから Java アプレットを削除する手順について説明します。Java アプレットは、保護されたネットワーク上のホストとサーバを攻撃するコードを含むことがあるため、セキュリティ リスクを引き起こす可能性があります。Java アプレットは、**filter java** コマンドで取り除くことができます。

filter java コマンドは、発信接続からセキュリティ アプライアンスに返される Java アプレットをフィルタリングします。フィルタリングされてもユーザは HTML ページを受信できますが、アプレットの Web ページ ソースはコメントアウトされているため、アプレットは実行できません。**filter java** コマンドでは、WebVPN トラフィックはフィルタリングされません。

（注） <object> タグに埋め込まれた Java アプレットを取り除くには、**filter activex** コマンドを使用します。

ファイアウォールを通過する HTTP トラフィック内の Java アプレットを取り除くには、グローバル コンフィギュレーション モードで次のコマンドを入力します。

```
hostname(config)# filter java  port[-port] local_ip local_mask foreign_ip foreign_mask
```

このコマンドを使用するには、*port* に、フィルタリングを適用する TCP ポートを指定します。一般的に、これはポート 80 ですが、他の値でも受け入れられます。ポート 80 には **http** または **url** リテラルを使用できます。ポート範囲を指定するには、開始ポート番号と終了ポート番号の間にハイフンを使用します。

ローカル IP アドレスおよびマスクによって、フィルタリングされるトラフィックの発信元である 1 台以上の内部ホストを指定します。外部アドレスおよびマスクは、フィルタリングされるトラフィックの外部の宛先を指定します。

これらのアドレスに **0.0.0.0**（短縮形は **0**）を設定して、すべてのホストを指定できます。これらのマスクに **0.0.0.0**（短縮形は **0**）を使用して、すべてのホストを指定できます。

これらのアドレスに **0.0.0.0**（短縮形は **0**）を設定して、すべてのホストを指定できます。これらのマスクに **0.0.0.0**（短縮形は **0**）を使用して、すべてのホストを指定できます。

次の例では、すべての発信接続で Java アプレットをブロックすることを指定しています。

```
hostname(config)# filter java 80 0 0 0 0
```

このコマンドは、Java アプレット ブロックが、あらゆるローカル ホストからあらゆる外部ホストへのポート 80 の Web トラフィックに対して、適用されることを指定しています。

次の例では、保護されたネットワーク上のホストへの Java アプレットのダウンロードをブロックしています。

```
hostname(config)# filter java http 192.168.3.3 255.255.255.255 0 0
```

このコマンドは、ホスト 192.168.3.3 による Java アプレットのダウンロードをブロックします。

このコンフィギュレーションを削除するには、次の例で示すように、コマンドの **no** 形式を使用します。

```
hostname(config)# no filter java http 192.168.3.3 255.255.255.255 0 0
```

外部サーバを使用した URL および FTP 要求のフィルタリング

この項では、外部サーバを使用して URL および FTP 要求をフィルタリングする方法について説明します。この項では、次のトピックについて取り上げます。

URL フィルタリングの概要

フィルタリングは、セキュリティの高いネットワークからセキュリティの低いネットワークに発信される接続要求に対して適用できます。ACL を使用して特定のコンテンツ サーバに対する発信アクセスを禁止することはできますが、サイズおよびインターネットのダイナミックな性質により、このような手段で使用方法を管理することは困難です。次のいずれかのインターネット フィルタリング製品で稼働する別途サーバを使用することで、設定を簡素化し、セキュリティ アプライアンス のパフォーマンスを向上できます。

- HTTP、HTTPS、および FTP フィルタリング用の Websense Enterprise
- HTTP、HTTPS、FTP、および長い URL フィルタリング用の Secure Computing SmartFilter（従来の N2H2）

（注） URL キャッシングが動作するのは、URL サーバのベンダーから提供された URL サーバ ソフトウェアのバージョンで URL キャッシングがサポートされている場合だけです。

外部サーバを使用するときは セキュリティ アプライアンス のパフォーマンスはほとんど影響を受けませんが、フィルタリング サーバが セキュリティ アプライアンス から離れた場所にある場合には、Web サイトまたは FTP サーバへのアクセス時間が大幅に長くなることがあります。

フィルタリングがイネーブルで、接続要求を セキュリティ アプライアンス 経由で転送すると、その要求はコンテンツ サーバとフィルタリング サーバに同時に送信されます。フィルタリング サーバによって接続が許可されると、セキュリティ アプライアンス はコンテンツ サーバからの応答を発信元のクライアントに転送します。フィルタリング サーバが接続を拒否した場合、セキュリティ アプライアンスは応答を廃棄し、接続が成功しなかったことを示すメッセージまたはリターン コードを送信します。

セキュリティ アプライアンス上でユーザ認証がイネーブルの場合、セキュリティ アプライアンスはフィルタリング サーバにユーザ名も送信します。フィルタリング サーバは、ユーザ固有のフィルタリング設定を使用したり、使用方法に関する高度なレポートを提供したりすることができます。

フィルタリング サーバの指定

コンテキストごとに最大 4 つのフィルタリング サーバを指定できます。セキュリティ アプライアンスは、1 つのサーバが応答するまで、それらのサーバを順番に使用します。コンフィギュレーション内に設定できるサーバのタイプは、1 つだけ（Websense または Secure Computing SmartFilter）です。

（注） **filter** コマンドを使用して HTTP または HTTPS のフィルタリングを設定する前に、フィルタリング サーバを追加する必要があります。コンフィギュレーションからフィルタリング サーバを削除すると、**filter** コマンドもすべて削除されます。

url-server コマンドを次のように使用して、フィルタリング サーバのアドレスを指定します。

Websense の場合は次のとおりです。

```
hostname(config)# url-server (if_name) host local_ip [timeout seconds] [protocol TCP | UDP
version [1|4] [connections num_conns] ]
```

Secure Computing SmartFilter（従来の N2H2）の場合は次のとおりです。

```
hostname(config)# url-server (if_name) vendor {secure-computing | n2h2} host
<local_ip> [port <number>] [timeout <seconds>] [protocol {TCP [connections <number>]} |
UDP]
```

<*if_name*> には、フィルタリング サーバに接続されているセキュリティ アプライアンス インターフェイスの名前を指定します（デフォルトは inside です）。

vendor {secure-computing | n2h2} には、ベンダー文字列として「secure-computing」を使用できますが、「n2h2」は下位互換性に使用できます。設定エントリが生成されるときに、「secure-computing」がベンダー文字列として保存されます。

host <*local_ip*> には、URL フィルタリング サーバの IP アドレスを指定します。

port <*number*> には、フィルタリング サーバの Secure Computing SmartFilter サーバ ポート番号を指定します。また、セキュリティ アプライアンスは、このポートの UDP 応答をリッスンします。

（注） デフォルト ポートは 4005 です。これは、Secure Computing SmartFilter サーバが TCP または UDP でセキュリティ アプライアンスと通信するために使用するデフォルト ポートです。デフォルト ポートの変更方法については、『*Filtering by N2H2 Administrator's Guide*』を参照してください。

timeout <*seconds*> は、セキュリティ アプライアンスがフィルタリング サーバへの接続試行を継続する秒数です。

connections <*number*> は、ホストとサーバの間で接続を試行する回数です。

たとえば、1 つの Websense フィルタリング サーバを指定するには、次のコマンドを入力します。

```
hostname(config)# url-server (perimeter) host 10.0.1.1 protocol TCP version 4
```

これは、セキュリティ アプライアンスの境界インターフェイス上の、IP アドレス 10.0.1.1 を持つ Websense フィルタリング サーバを指定しています。この例でイネーブルになっている version 4 は、キャッシュをサポートするため、Websense によって推奨されています。

冗長 Secure Computing SmartFilter サーバを指定するには、次のコマンドを入力します。

```
hostname(config)# url-server (perimeter) vendor n2h2 host 10.0.1.1
hostname(config)# url-server (perimeter) vendor n2h2 host 10.0.1.2
```

これは、2 つの Sentian フィルタリング サーバを指定しています。どちらもセキュリティ アプライアンスの境界インターフェイス上にあります。

コンテンツ サーバ応答のバッファリング

ユーザがコンテンツ サーバへの接続要求を発行した場合、その要求は、セキュリティ アプライアンスによって、コンテンツ サーバとフィルタリング サーバの両方に同時に送信されます。フィルタリング サーバがコンテンツ サーバより早く応答しなかった場合、サーバ応答はドロップされます。これにより、Web クライアント側の視点で Web サーバ応答が表示されます。これは、クライアントが要求を再発行する必要があるためです。

HTTP 応答バッファをイネーブルにすると、Web コンテンツ サーバからの応答はバッファリングされ、フィルタリング サーバによって接続が許可された場合に、要求クライアントに転送されます。これにより、バッファリングしない場合に発生する可能性のある遅延が回避されます。

HTTP 要求または FTP 要求に対する応答のバッファリングを設定するには、次の手順を実行します。

ステップ 1　フィルタリング サーバからの応答が保留中である HTTP または FTP 要求に対する応答のバッファリングをイネーブルにするには、次のコマンドを入力します。

```
hostname(config)# url-block block block-buffer-limit
```

block-buffer に、url-server からの応答を待っている間にバッファリング可能な HTTP 応答の最大数を指定します。

（注）　3072 バイトより長い URL のバッファリングはサポートされていません。

ステップ 2　次のコマンドを入力して、保留中 URL バッファリング（および長い URL バッファリング）用の最大使用可能メモリを設定します。

```
hostname(config)# url-block mempool-size memory-pool-size
```

最大メモリ割り当ての 2 KB ～ 10 MB に相当する 2 ～ 10240 の範囲の値を、memory-pool-size に指定します。

サーバ アドレスのキャッシング

ユーザがサイトにアクセスすると、フィルタリング サーバはセキュリティ アプライアンスに対して、サーバ アドレスを一定時間キャッシュすることを許可できます。ただし、そのアドレスでホストされているサイトはいずれも、常に許可されるカテゴリに属している必要があります。これにより、そのユーザがそのサーバに再度アクセスするか、別のユーザがそのサーバにアクセスしたときに、セキュリティ アプライアンスがフィルタリング サーバに再度照会する必要がなくなります。

（注）　キャッシュされた IP アドレス要求は、フィルタリング サーバに渡されず、記録もされません。そのため、このアクティビティはどのレポートにも表示されません。**url-cache** コマンドを使用する前に、Websense 実行ログを蓄積できます。

スループットを高める必要がある場合は、**url-cache** コマンドを使用して、次のように入力します。

```
hostname(config)# url-cache dst | src_dst size
```

範囲 1 ～ 128（KB）のキャッシュ サイズの値を、*size* に指定します。

dst キーワードを使用して、URL 宛先アドレスに基づいて、エントリをキャッシュします。このモードは、Websense サーバ上ですべてのユーザが同じ URL フィルタリング ポリシーを共有する場合に選択します。

src_dst キーワードを使用して、URL 要求を開始した送信元アドレスと URL 宛先アドレスの両方に基づいて、エントリをキャッシュします。このモードは、Websense サーバ上でユーザが同じ URL フィルタリング ポリシーを共有しない場合に選択します。

HTTP URL のフィルタリング

この項では、外部フィルタリング サーバを使用する HTTP フィルタリングを設定する方法について説明します。この項では、次のトピックについて取り上げます。

- 「HTTP フィルタリングの設定」（P.20-7）
- 「長い HTTP URL のフィルタリングのイネーブル化」（P.20-8）
- 「長い HTTP URL の切り捨て」（P.20-8）
- 「トラフィックに対するフィルタリングの免除」（P.20-8）

HTTP フィルタリングの設定

HTTP フィルタリングをイネーブルにする前に、URL フィルタリング サーバを指定し、イネーブルにする必要があります。

フィルタリング サーバが HTTP 接続要求を承認した場合、セキュリティ アプライアンスは Web サーバからの応答が発信元クライアントに到達することを許可します。フィルタリング サーバが要求を拒否した場合、セキュリティ アプライアンスは、ユーザをブロック ページにリダイレクトし、アクセスが拒否されたことを示します。

HTTP フィルタリングをイネーブルにするには、次のコマンドを入力します。

```
hostname(config)# filter url [http | port[-port] local_ip local_mask foreign_ip
foreign_mask] [allow] [proxy-block]
```

HTTP（80）のデフォルト ポートとは異なるポートが使用されている場合は、1 つまたは複数のポート番号を、*port* に指定します。*local_ip* と *local_mask* には、要求を行うユーザまたはサブネットワークの IP アドレスとサブネット マスクを指定します。*foreign_ip* と *foreign_mask* には、要求に応答するサーバまたはサブネットワークの IP アドレスとサブネット マスクを指定します。

allow オプションは、プライマリ フィルタリング サーバが利用できないときに、セキュリティ アプライアンスがフィルタリングせずに HTTP トラフィックを転送するようにします。**proxy-block** コマンドを使用して、プロキシ サーバへの要求をすべてドロップします。

長い HTTP URL のフィルタリングのイネーブル化

デフォルトでは、セキュリティ アプライアンスは、1159 文字を超える HTTP URL を長い URL と見なします。最大許容量を大きくすることができます。

次のコマンドを使用して、1 つの URL の最大サイズを設定します。

```
hostname(config)# url-block url-size long-url-size
```

long-url-size には、バッファリングされる長い URL それぞれの最大サイズ（KB）を指定します。Websense の場合、この値は 2 ～ 4 で最大 URL サイズは 2 KB ～ 4 KB となります。Secure Computing の場合、この値は 2 ～ 3 で最大 URL サイズは 2 KB ～ 3 KB となります。デフォルト値は 2 です。

長い HTTP URL の切り捨て

デフォルトでは、URL が最大許容サイズを超えると、その URL はドロップされます。これを回避するには、次のコマンドを入力して、長い URL を切り捨てるようにセキュリティ アプライアンスを設定します。

```
hostname(config)# filter url [longurl-truncate | longurl-deny | cgi-truncate]
```

longurl-truncate オプションを指定すると、セキュリティ アプライアンスは URL が最大許容長よりも長い場合に、URL のホスト名または IP アドレス部分だけを、評価のためにフィルタリング サーバに送信します。**longurl-deny** オプションは、URL が最大許容長よりも長い場合、発信 URL トラフィックを拒否します。

パラメータは含まずに CGI スクリプトの場所とスクリプト名だけを含むよう CGI URL を切り捨てるには、**cgi-truncate** オプションを使用します。長い HTTP 要求のほとんどは、CGI 要求です。パラメータ リストが非常に長い場合、パラメータ リストを含む完全な CGI 要求を待機および送信すると、メモリ リソースが浪費され、ファイアウォールのパフォーマンスに影響します。

トラフィックに対するフィルタリングの免除

フィルタリングから除外する特定のトラフィックを指定するには、次のコマンドを入力します。

```
hostname(config)# filter url except source_ip source_mask dest_ip dest_mask
```

たとえば、次のコマンドは、10.0.2.54 からの HTTP 要求を除くすべての HTTP 要求がフィルタリング サーバに転送されるように設定しています。

```
hostname(config)# filter url http 0 0 0 0
hostname(config)# filter url except 10.0.2.54 255.255.255.255 0 0
```

HTTPS URL のフィルタリング

HTTPS フィルタリングをイネーブルにする前に、URL フィルタリング サーバを指定し、イネーブルにする必要があります。

（注） 現在、Websense および Smartfilter は HTTPS をサポートしています。古いバージョンの Secure Computing SmartFilter（従来の N2H2）では HTTPS のフィルタリングをサポートしていませんでした。

HTTPS コンテンツは暗号化されているため、セキュリティ アプライアンスは、ディレクトリおよびファイル名の情報を付けずに URL ルックアップを送信します。フィルタリング サーバが HTTPS 接続要求を承認した場合、セキュリティ アプライアンスは SSL 接続ネゴシエーションの完了を許可し、

Web サーバからの応答が発信元クライアントに到達することを許可します。フィルタリング サーバが要求を拒否した場合、セキュリティ アプライアンスは SSL 接続ネゴシエーションの完了を許可しません。ブラウザには、「The Page or the content cannot be displayed.」のようなエラー メッセージが表示されます。

（注）　セキュリティ アプライアンスは、HTTPS 用の認証プロンプトを表示しないため、ユーザは HTTPS サーバにアクセスする前に、HTTP または FTP を使用してセキュリティ アプライアンスで認証を受ける必要があります。

HTTPS フィルタリングをイネーブルにするには、次のコマンドを入力します。

```
hostname(config)# filter https port[-port] localIP local_mask foreign_IP foreign_mask
[allow]
```

HTTPS（443）のデフォルト ポートとは異なるポートが使用されている場合は、ポート番号の範囲を *port*[*-port*] に指定します。

local_ip と *local_mask* には、要求を行うユーザまたはサブネットワークの IP アドレスとサブネット マスクを指定します。

foreign_ip と *foreign_mask* には、要求に応答するサーバまたはサブネットワークの IP アドレスとサブネット マスクを指定します。

allow オプションは、プライマリ フィルタリング サーバが利用できないときに、セキュリティ アプライアンスがフィルタリングせずに HTTPS トラフィックを転送するようにします。

FTP 要求のフィルタリング

FTP フィルタリングをイネーブルにする前に、URL フィルタリング サーバを指定し、イネーブルにする必要があります。

（注）　現在、Websense および Smartfilter は FTP をサポートしています。古いバージョンの Secure Computing SmartFilter（従来の N2H2）では、FTP のフィルタリングをサポートしていませんでした。

フィルタリング サーバが FTP 接続要求を承認した場合、セキュリティ アプライアンスは、成功を示す FTP リターン コードが発信元クライアントに到達することを許可します。たとえば、成功を示すリターン コードは「250: CWD command successful」です。フィルタリング サーバが要求を拒否した場合、FTP リターン コードは接続が拒否されたことを示すように変更されます。たとえば、セキュリティ アプライアンスの場合、コード 250 を「550 Requested file is prohibited by URL filtering policy」に変更します。

FTP フィルタリングをイネーブルにするには、次のコマンドを入力します。

```
hostname(config)# filter ftp port[-port] localIP local_mask foreign_IP foreign_mask
[allow] [interact-block]
```

FTP（21）のデフォルト ポートとは異なるポートが使用されている場合は、ポート番号の範囲を *port*[*-port*] に指定します。

local_ip と *local_mask* には、要求を行うユーザまたはサブネットワークの IP アドレスとサブネット マスクを指定します。

foreign_ip と *foreign_mask* には、要求に応答するサーバまたはサブネットワークの IP アドレスとサブネット マスクを指定します。

allow オプションは、プライマリ フィルタリング サーバが利用できないときに、セキュリティ アプライアンスがフィルタリングせずに HTTPS トラフィックを転送するようにします。

完全なディレクトリ パスを提供しない対話型の FTP セッションをブロックするには、**interact-block** オプションを使用します。対話形式の FTP クライアントを使用すると、ユーザは、完全なパスを入力しないでディレクトリを変更できます。たとえば、ユーザは、**cd /public/files** ではなく、**cd ./files** と入力できます。

フィルタリング統計情報とフィルタリング設定の表示

この項では、フィルタリング統計情報をモニタする方法について説明します。この項では、次のトピックについて取り上げます。

フィルタリング サーバ統計情報の表示

フィルタリング サーバの情報を表示するには、次のコマンドを入力します。

```
hostname# show running-config url-server
```

次に、**show running-config url-server** コマンドの出力例を示します。

```
hostname# show running-config url-server
url-server (outside) vendor n2h2 host 128.107.254.202 port 4005 timeout 5 protocol TCP
```

フィルタリング サーバの情報または統計情報を表示するには、次のコマンドを入力します。

次に、**show running-config url-server statistics** コマンドの出力例を示します。このコマンドでは、フィルタリング統計情報が表示されます。

```
hostname# show running-config url-server statistics

Global Statistics:
--------------------
URLs total/allowed/denied         13/3/10
URLs allowed by cache/server      0/3
URLs denied by cache/server       0/10
HTTPSs total/allowed/denied       138/137/1
HTTPSs allowed by cache/server    0/137
HTTPSs denied by cache/server     0/1
FTPs total/allowed/denied         0/0/0
FTPs allowed by cache/server      0/0
FTPs denied by cache/server       0/0
Requests dropped                  0
Server timeouts/retries           0/0
Processed rate average 60s/300s   0/0 requests/second
Denied rate average 60s/300s      0/0 requests/second
Dropped rate average 60s/300s     0/0 requests/second

Server Statistics:
--------------------
```

```
10.125.76.20                        UP
  Vendor                            websense
  Port                              15868
  Requests total/allowed/denied     151/140/11
  Server timeouts/retries           0/0
  Responses received                151
  Response time average 60s/300s    0/0

URL Packets Sent and Received Stats:
------------------------------------
Message                  Sent    Received
STATUS_REQUEST           1609    1601
LOOKUP_REQUEST           1526    1526
LOG_REQUEST              0       NA

Errors:
-------
RFC noncompliant GET method     0
URL buffer update failure       0
```

バッファ コンフィギュレーションと統計情報の表示

show running-config url-block コマンドは、url-block バッファで保持されるパケット数と、バッファ制限を超えた場合または再送信が発生した場合に廃棄される数（存在する場合）を示します。

次に、**showrunning-configurl-block** コマンドの出力例を示します。

```
hostname# show running-config url-block
    url-block url-mempool 128
    url-block url-size 4
    url-block block 128
```

URL ブロック バッファのコンフィギュレーションが表示されています。

次に、**show url-block block statistics** コマンドの出力例を示します。

```
hostname# show running-config url-block block statistics

URL Pending Packet Buffer Stats with max block  128
-----------------------------------------------------
Cumulative number of packets held:              896
Maximum number of packets held (per URL):       3
Current number of packets held (global):        38
Packets dropped due to
       exceeding url-block buffer limit:        7546
       HTTP server retransmission:              10
Number of packets released back to client:      0
```

これは、URL ブロック統計情報を示しています。

キャッシュ統計情報の表示

次に、**show url-cache stats** コマンドの出力例を示します。

```
hostname# show url-cache stats
URL Filter Cache Stats
----------------------
    Size :     128KB
 Entries :      1724
  In Use :       456
```

```
 Lookups :         45
    Hits :          8

This shows how the cache is used.
```

フィルタリング性能統計情報の表示

次に、**show perfmon** コマンドの出力例を示します。

```
hostname# show perfmon
PERFMON STATS:    Current      Average
Xlates               0/s          0/s
Connections          0/s          2/s
TCP Conns            0/s          2/s
UDP Conns            0/s          0/s
URL Access           0/s          2/s
URL Server Req       0/s          3/s
TCP Fixup            0/s          0/s
TCPIntercept         0/s          0/s
HTTP Fixup           0/s          3/s
FTP Fixup            0/s          0/s
AAA Authen           0/s          0/s
AAA Author           0/s          0/s
AAA Account          0/s          0/s
```

これは、URL フィルタリング性能統計情報とその他の性能統計情報を示しています。フィルタリング統計情報は URL Access 行および URL Server Req 行に表示されます。

フィルタリング コンフィギュレーションの表示

次に、**show running-config filter** コマンドの出力例を示します。

```
hostname# show running-config filter
filter url http 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
```