COPYRIGHT
Copyright © 2015 McAfee, Inc., 2821 Mission College Boulevard, Santa Clara, CA 95054, 1.888.847.8766, www.intelsecurity.com

TRADEMARK ATTRIBUTIONS
Intel and the Intel logo are registered trademarks of the Intel Corporation in the US and/or other countries. McAfee and the McAfee logo, McAfee Active Protection, McAfee DeepSAFE, ePolicy Orchestrator, McAfee ePO, McAfee EMM, McAfee Evader, Foundscore, Foundstone, Global Threat Intelligence, McAfee LiveSafe, Policy Lab, McAfee QuickClean, Safe Eyes, McAfee SECURE, McAfee Shredder, SiteAdvisor, McAfee Stinger, McAfee TechMaster, McAfee Total Protection, TrustedSource, VirusScan are registered trademarks or trademarks of McAfee, Inc. or its subsidiaries in the US and other countries. Other marks and brands may be claimed as the property of others.

LICENSE INFORMATION
License Agreement
NOTICE TO ALL USERS: CAREFULLY READ THE APPROPRIATE LEGAL AGREEMENT CORRESPONDING TO THE LICENSE YOU PURCHASED, WHICH SETS FORTH THE GENERAL TERMS AND CONDITIONS FOR THE USE OF THE LICENSED SOFTWARE. IF YOU DO NOT KNOW WHICH TYPE OF LICENSE YOU HAVE ACQUIRED, PLEASE CONSULT THE SALES AND OTHER RELATED LICENSE GRANT OR PURCHASE ORDER DOCUMENTS THAT ACCOMPANY YOUR SOFTWARE PACKAGING OR THAT YOU HAVE RECEIVED SEPARATELY AS PART OF THE PURCHASE (AS A BOOKLET, A FILE ON THE PRODUCT CD, OR A FILE AVAILABLE ON THE WEBSITE FROM WHICH YOU DOWNLOADED THE SOFTWARE PACKAGE). IF YOU DO NOT AGREE TO ALL OF THE TERMS SET FORTH IN THE AGREEMENT, DO NOT INSTALL THE SOFTWARE. IF APPLICABLE, YOU MAY RETURN THE PRODUCT TO McAfee OR THE PLACE OF PURCHASE FOR A FULL REFUND.
# Preface

- About this guide
- Audience
- Conventions

# Custom attacks

- Reasons to create your own attack definitions
- Types of custom attacks
- Network Security Platform signature terminology
  - Signatures and rules
  - Attacks, policies, and alerts
  - Custom attack editor

# Creating custom attacks

- Before you create a custom attack
- Required information for creating a custom attack
- Understanding impact packages and protocols
  - How Network Security Platform prevents intrusions
  - How attack detection works
- Technical information references
  - Importance of testing custom attack definitions

# Quick tour of the custom attack editor

- Basics of the custom attack editor interface
  - Menu items
  - Tab regions
  - View options
  - Create a display filter
  - Attack creation interfaces
    - McAfee custom attack creation interfaces
    - Snort custom attack interfaces
  - Test compile custom attacks

# Mechanics of a custom attack

- Structure of a custom attack
  - Structure of a signature
  - Create more than one signature per attack
- Signature test reference
  - Numeric comparison tests
  - Numeric range tests
  - Numeric enumeration tests
  - Pattern matching
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a fixed field signature</td>
<td>39</td>
</tr>
<tr>
<td>Packet search protocols</td>
<td>39</td>
</tr>
<tr>
<td>Performance issues</td>
<td>40</td>
</tr>
<tr>
<td>Considerations</td>
<td>40</td>
</tr>
<tr>
<td>5 Managing custom attacks</td>
<td>41</td>
</tr>
<tr>
<td>Create custom attacks</td>
<td>41</td>
</tr>
<tr>
<td>Create an exploit attack without template</td>
<td>42</td>
</tr>
<tr>
<td>Create a signature</td>
<td>44</td>
</tr>
<tr>
<td>Templates for McAfee custom attacks</td>
<td>47</td>
</tr>
<tr>
<td>Configure custom reconnaissance attack definition</td>
<td>61</td>
</tr>
<tr>
<td>Configure custom attack definition for multiple attacks correlation</td>
<td>64</td>
</tr>
<tr>
<td>Import previously exported custom attacks</td>
<td>64</td>
</tr>
<tr>
<td>Importing McAfee-defined custom attacks</td>
<td>65</td>
</tr>
<tr>
<td>Work with packet search protocol</td>
<td>65</td>
</tr>
<tr>
<td>Create a packet search protocol</td>
<td>66</td>
</tr>
<tr>
<td>Create a packet search protocol instance</td>
<td>67</td>
</tr>
<tr>
<td>Create a signature using your new packet search protocol</td>
<td>68</td>
</tr>
<tr>
<td>Regular expression language</td>
<td>70</td>
</tr>
<tr>
<td>Regular expression syntax</td>
<td>71</td>
</tr>
<tr>
<td>Limitations of a custom reconnaissance attack</td>
<td>73</td>
</tr>
<tr>
<td>6 Mechanics of a Snort custom attack</td>
<td>75</td>
</tr>
<tr>
<td>Structure of a Snort custom attack</td>
<td>75</td>
</tr>
<tr>
<td>Structure of a snort rule</td>
<td>76</td>
</tr>
<tr>
<td>Rule header section</td>
<td>76</td>
</tr>
<tr>
<td>Rule options section</td>
<td>78</td>
</tr>
<tr>
<td>7 Management of snort custom attacks</td>
<td>107</td>
</tr>
<tr>
<td>Considerations</td>
<td>107</td>
</tr>
<tr>
<td>Best practices</td>
<td>110</td>
</tr>
<tr>
<td>Create snort custom attacks</td>
<td>111</td>
</tr>
<tr>
<td>Variables</td>
<td>111</td>
</tr>
<tr>
<td>Define the snort variables</td>
<td>112</td>
</tr>
<tr>
<td>Define the classification types</td>
<td>113</td>
</tr>
<tr>
<td>View the variables and classification types</td>
<td>113</td>
</tr>
<tr>
<td>Identification of the protocol of a snort custom attack</td>
<td>115</td>
</tr>
<tr>
<td>How to use snort rules to detect IP communication between specific hosts</td>
<td>116</td>
</tr>
<tr>
<td>Importing snort custom attacks</td>
<td>117</td>
</tr>
<tr>
<td>Snort rule validation utility</td>
<td>118</td>
</tr>
<tr>
<td>Include duplicate Snort custom attacks</td>
<td>118</td>
</tr>
<tr>
<td>Import snort rules through a conf file</td>
<td>119</td>
</tr>
<tr>
<td>Import snort rules through a rules file</td>
<td>121</td>
</tr>
<tr>
<td>Viewing the imported Snort Custom Attacks</td>
<td>122</td>
</tr>
<tr>
<td>Create a display filter</td>
<td>124</td>
</tr>
<tr>
<td>Correct the failed conversions</td>
<td>125</td>
</tr>
<tr>
<td>Write snort custom attacks</td>
<td>126</td>
</tr>
<tr>
<td>Saving the Snort custom attacks</td>
<td>127</td>
</tr>
<tr>
<td>Save the qualified rules</td>
<td>128</td>
</tr>
<tr>
<td>Customizing the snort rules attack responses</td>
<td>128</td>
</tr>
<tr>
<td>Delete the snort rules from the database</td>
<td>129</td>
</tr>
<tr>
<td>8 Common tasks</td>
<td>131</td>
</tr>
<tr>
<td>How attacks are included in policies</td>
<td>131</td>
</tr>
<tr>
<td>Viewing a policy to verify inclusion of the attack</td>
<td>132</td>
</tr>
<tr>
<td>Verify the inclusion of custom attack in a rule set</td>
<td>133</td>
</tr>
</tbody>
</table>
Contents

Add attack descriptions to the Attack Encyclopedia ................................................. 134
Compile the attack definitions .................................................................................. 137
Update the Sensor configuration to apply a policy .................................................. 137
Custom attacks export .............................................................................................. 138
Export the custom attacks ......................................................................................... 139

9 Examples .............................................................................................................. 141
Custom attack with a pattern-match signature ......................................................... 141
  Create a custom attack .......................................................................................... 141
  Create a snort custom attack — an example ....................................................... 145
Custom attacks with multiple comparisons ............................................................ 148
  Create an attack definition for the example ....................................................... 148
  Equivalent snort custom attack ......................................................................... 153

10 Management of custom attacks from the Central Manager ......................... 155
Important notes ........................................................................................................ 155

Index ....................................................................................................................... 157
Preface

This guide provides the information you need to configure, use, and maintain your McAfee product.

Contents

- About this guide
- Find product documentation

About this guide

This information describes the guide's target audience, the typographical conventions and icons used in this guide, and how the guide is organized.

Audience

McAfee documentation is carefully researched and written for the target audience.

The information in this guide is intended primarily for:

- **Administrators** — People who implement and enforce the company's security program.
- **Users** — People who use the computer where the software is running and can access some or all of its features.

Conventions

This guide uses these typographical conventions and icons.

- **Book title, term, emphasis**
  
  Title of a book, chapter, or topic; a new term; emphasis.

- **Bold**
  
  Text that is strongly emphasized.

- **User input, code, message**
  
  Commands and other text that the user types; a code sample; a displayed message.

- **Interface text**
  
  Words from the product interface like options, menus, buttons, and dialog boxes.

- **Hypertext blue**
  
  A link to a topic or to an external website.

- **Note:** Additional information, like an alternate method of accessing an option.

- **Tip:** Suggestions and recommendations.

- **Important/Caution:** Valuable advice to protect your computer system, software installation, network, business, or data.

- **Warning:** Critical advice to prevent bodily harm when using a hardware product.
Find product documentation

After a product is released, information about the product is entered into the McAfee online Knowledge Center.

**Task**

1. Go to the **Knowledge Center** tab of the McAfee ServicePortal at [http://support.mcafee.com](http://support.mcafee.com).

2. In the **Support Content** pane:
   - Click **Product Documentation** to find user documentation.
   - Click **Technical Articles** to find KnowledgeBase articles.

3. Select **Do not clear my filters**.

4. Enter a product, select a version, then click **Search** to display a list of documents.
McAfee® Network Security Platform [formerly McAfee® IntruShield®] uses a variety of methods to detect aberrant network activity. The main detection method used is attack definition. An attack defines either a rule or related signatures that specify a set of checks for capturing various attempts at exploiting a given vulnerability or creating other threat conditions.

Throughout this guide, the terms attack and attack definition are used interchangeably.

McAfee regularly supplies you with its own attack definitions (signature set) to protect your network. Additionally, it also provides ad hoc signature sets in case of emergencies and zero-day vulnerabilities. McAfee’s research team develops these signatures and tests them thoroughly before releasing them to its customers.

There could be unique security requirements that would not be possible to be covered in the McAfee-supplied signature set. For such cases, you have the option of developing your own attack definitions. Such user-defined attacks are referred to as custom attack definitions or custom attacks.

### Contents
- Reasons to create your own attack definitions
- Types of custom attacks
- Network Security Platform signature terminology
- Custom attack editor

### Reasons to create your own attack definitions

If the McAfee-supplied signature set is adequate to protect your network, then why create one yourself?

The reasons you might want to create an attack will vary widely based on the usage scenario for your McAfee® Network Security Sensor and the environment into which it is deployed. The sections below attempt to describe some common scenarios and the actions that might be taken when they occur.

- **Emergency Situations** - McAfee follows strict quality assurance processes and requires high-quality information before providing updates to customers. Thus, there may be situations where an attack covering particular vulnerabilities may be a very high priority for your network but might not be provided as swiftly as your security policy might require. In this case, you might need to write and deploy strategic attacks based on your knowledge of your network's vulnerabilities until an update from McAfee is available.

- **Forensic Analysis** - It may be useful to deploy specialized attacks to investigate suspicious activity on your network and possibly to track intrusions.
• **Custom attacks for Your Particular Environment** - Writing custom attacks for your network environment may be necessary for a variety of reasons. The most common is policy enforcement. For example, your security policy may dictate that certain traffic or usage patterns are disallowed. In such cases, it may be useful to write and deploy attacks that can alert your IT or security staff.

• **To preserve legacy attack definitions** - If you are migrating from an open-source IPS or IDS solution, such as Snort, you may want to import the attacks in their current format into Network Security Platform.

## Types of custom attacks

In Network Security Platform, you have the flexibility to create custom attacks in two ways:

• You can create custom attacks in McAfee's proprietary format. This type of custom attacks are signature-based. You can define one or more signatures per attack. Such attack definitions are called **McAfee Custom Attacks** in this guide.

  In the earlier releases, the McAfee Custom Attacks was called User-Defined Signatures (UDS). The interfaces of the UDS Editor are now available within the Custom Attack Editor.

• You can write rule-based custom attacks using Snort rules language, which is open-source. Such attack definitions are called **Snort Custom Attacks** in this guide.

  These two formats are not interchangeable. That is, you cannot convert a McAfee Custom Attack to Snort Custom Attack or the other way around.

You can use Snort rules in your existing Network Security Platform setup without having to modify it in any way or install any Snort-related components. These rules could be your own or from sources such as the Snort user-community; the critical thing is that the rules should conform to the Snort syntax for you to use them in Network Security Platform.

A Snort Custom Attack is converted into McAfee's format internally when you save it in the Manager server. This translation enables you to use Snort rules directly in Network Security Platform without the need for any Snort-related components.

## Network Security Platform signature terminology

To be able to create effective custom attacks, you need to understand how Network Security Platform detects attacks. To be able to understand this mechanism, you need to be familiar with some terms listed below:

• Signatures and rules
• Attacks
• Policies
• Alerts

### Signatures and rules

A signature or rule is a set of checks (for example, string matches or IP-port comparisons) that are applied to network traffic seen by the Sensor. The term signature is relevant for McAfee Custom Attacks (that is, custom attacks in McAfee format) or the McAfee-supplied signature set. The term rule applies to Snort Custom Attacks.
In case of Snort Custom Attacks, the Manager parses them to check the syntax. Then it converts the valid attack definitions to McAfee's format and saves them in the Manager database. Once in the database, the converted Snort rules function like any other Network Security Platform signature. So, in Network Security Platform, both signatures and rules are functionally similar. What applies to a signature is also applicable to a rule unless stated otherwise.

Throughout this guide, Snort Custom Attacks are also referred to as Snort rules or just rules. This is not to be confused with the Network Security Platform rule sets, which are a collection of attack definitions that meet certain criteria.

The attack detection mechanism in Network Security Platform relies on comparison of traffic to a database of signatures or rules. Network Security Platform enables you to define checks using combinations of string matches and checks for other anomalies such as excessive field lengths. When all the necessary conditions for a given rule of set of signatures are satisfied, an event is raised in the Sensor; all signatures and rules defined are checked against the traffic simultaneously and all matching events within the context of a given attack are correlated by the Sensor to generate a single alert.

**Attacks, policies, and alerts**

In Network Security Platform, attack definitions are a mechanism used to identify and protect against malicious actions taken against your network. An attack definition is the aggregation of the signatures (or rule) and other supporting data that can identify a specific network event. When you select an attack, you are essentially selecting a group of conditions defined in a rule or signature(s).

A rule in a Snort Custom Attack corresponds to the signatures in a McAfee Custom Attack.

Policies are applied to the Sensor and consist of one or more attacks. Traffic passing through the Sensor is compared to the attacks enabled in the policies, and if any traffic is identified as malicious by an attack definition, an alert is triggered to notify you of the incident.

**Custom attack editor**

Custom Attack Editor provided in the Manager is the tool that you use to create custom attacks. Custom Attack Editor enables you to create McAfee Custom Attacks as well as in Snort Custom Attacks. Using this tool, you can also import custom attacks in bulk against defining them individually.

**Audience**

Custom Attack Editor is designed for sophisticated users with expertise in networking and intrusion prevention. The ability to create a custom attack is a double-edged sword. It can be a very powerful detection and defense mechanism, but at the same time, used without training and experience, a custom attack can cause harm to your network and the business that depends on it. In addition, without a significant amount of experience both using and configuring detection mechanisms for network intrusion detection devices, it is possible to make mistakes that can render your detection device essentially useless. For example, a mistake in implementation of a signature on a high-traffic network could cause such a large number of alerts to be generated that it would render the Manager unusable. On the opposite end of the spectrum, without proper experience and expertise, it is likely that a user might create a signature that would never detect (due to errors in tests or detection window, for example), despite the fact that the signature might be intended to detect very important events specific to your network.
Since a poorly written custom attack can cause many more problems than it solves, McAfee recommends that an attack writer possess the following knowledge:

- A strong understanding of computer networking.
- Experience with networks running the protocol for which you intend to create a custom attack, including a good packet-level understanding of the protocol.
- The ability to recognize the difference between "good" and "bad" traffic. That is, traffic that is correct and valid for your network and the devices that comprise it, and traffic that is anomalous to your network’s configuration and security policies.
- A strong understanding of Snort rules language, if you plan to use Snort Custom Attacks.

While the fact that Network Security Platform provides deep parsing of application protocols and supports complex attack-definition structures may seem overwhelming to first-time users, it is reasonably straightforward to create custom attacks after crossing this initial hurdle.
Creating custom attacks

Creating a custom attack requires certain experience as well as a number of different pieces of information, both technical and policy-oriented. This chapter describes the information you should have before you create custom attacks in Network Security Platform.

See also
Create custom attacks on page 41

Contents
- Before you create a custom attack
- Required information for creating a custom attack
- Understanding impact packages and protocols
- How Network Security Platform prevents intrusions
- Technical information references
- Importance of testing custom attack definitions

Before you create a custom attack

Creating attack definitions is a complex topic on which books have been written. It is highly recommended that if you are not an experienced attack definition author you refer to one of these books. Pay careful attention when you define your attacks, as there are many ways to shoot yourself (and your network!) in the foot as you develop expertise with attack definitions.

Before creating a custom attack, you should carefully consider the requirement that you are trying to address. This means that you should have a clear understanding of the attack's purpose, such as a specific business or policy requirement.

You should have a clear rationale for using a custom attack instead of another mechanism. In some cases, a custom attack may not be the appropriate solution. For this reason, you should consider whether you can address your need with other technical means, such as router ACLs, firewall rules, or a network sniffer.

Finally, you should verify that the custom attack you intend to create does not duplicate any attack provided in Network Security Platform. However, in case of a duplicate, you have the flexibility to use both or just the custom attack instead of the McAfee-supplied attack. If you choose to use both, then note that the Sensor may raise two alerts for the same attack traffic.
Required information for creating a custom attack

The following is a list of the information you should have on hand as you create your custom attacks and the constituent signatures or rule. A signature or rule can range from very simple - checking the value of a header field - to highly complex checks of different information in a specific order. You must have a good bit of data to aid you in creating an accurate attack definition.

- Reason for creating this custom attack.
- Technical information references for this custom attack.
- Protocol in which this custom attack will search the traffic (also known as the impact protocol).
- Specific hardware or software platforms affected by this traffic (also known as impact packages).
- Severity of this event.
- The direction in which the "traffic to be watched for" occurs.
- Specific criteria that comprise the attack, such as field values and patterns to match.
- A method, data, or tool to be used for testing the attack before you use it in your production environment.

Understanding impact packages and protocols

Network Security Platform policies define what hardware and software platforms are present on a network and which should be monitored. The "platforms" can be as generic as the HTTP protocol or as specific as Microsoft Internet Explorer running on Windows XP.

After a policy is configured and saved, the Manager searches through the database and generates a list of attacks that will be enabled when that policy is applied to an interface. Attacks specify what platforms they affect as part of their impact construct.

In case of McAfee Custom Attacks, the impact construct of an attack can contain references to protocols, and the packages that implement support for those protocols. Protocol references refer to protocols defined by McAfee as well as the custom-defined ones. Package references refer to particular software or hardware platforms, which can include a specific OS.

The platforms that can be selected are those for which Network Security Platform has specific support in some form. If you do not see a particular platform, you can choose the one which is similar, or specify "tcpip-machine" as the package, or specify just the impact protocol for the attack.

For Snort Custom Attacks, you cannot choose an impact package. By default, "tcpip-machine" is selected as the package, and this cannot be modified. Additionally, the Manager identifies the impact protocol for the attack definition.

How Network Security Platform prevents intrusions

Network Security Platform’s intrusion prevention mechanism is very powerful, but may also be difficult to understand. A good analogy is to that of a DNA test. DNA testing allows biology experts to obtain a DNA sample from a member of a species, and use the sample to determine the precise individual from which the particular sample came. Network Security Platform provides much the same capability, but oriented toward detecting and accurately identifying network events. As an example, Network Security Platform’s detection mechanisms can allow your signature to identify every HTTP traffic flow, every
HTTP traffic flow using the GET mechanism, every HTTP traffic flow using GET with `/cgi-bin/calendar.pl` as the path and even every GET with that path and a parameter named `month` with a value of February.

This is why Network Security Platform supports the aggregation of multiple conditions into every attack. Each signature or rule within an attack can be more or less specific so as to identify everything from generic network activity that affects a given platform in a particular way to a specific piece of code that has very specific and identifiable effects. Based on their specificity and severity, signatures and rules are assigned different confidence and severity values.

When a network event occurs that matches an existing attack definition, the rule or signature(s) (generic and specific) within that attack definition may be triggered. When alert throttling is enabled, the Sensor correlates the multiple triggering events automatically to raise a single alert with the highest confidence level.

### How attack detection works

The McAfee® Network Security Sensor (Sensor) performs different levels of traffic processing and analysis. Signatures and rules operate on traffic that has passed through these higher-level processing phases. Understanding how attack detection works can enable you to create effective Custom Attacks.

#### Flows

At the highest level, the Sensor addresses UDP and TCP traffic based on the concept of a flow. Flows are defined by their protocol (UDP/TCP) and the source and destination ports and IP addresses of their endpoints. As you are probably aware, UDP does not contain the concept of "state" that TCP does, so the Sensor implements a timer-based flow context for UDP traffic. After dividing traffic into flows, the Sensor makes use of port mappings or, in the case of traffic running on non-standard ports, intelligent protocol identification, to pass each flow to the appropriate protocol parsing mechanism.

For a custom attack, you can specify whether the Sensor should look at the complete flow, one direction of the flow, or restrict itself to data occurring within single packets of the flow. Precise control of this detection window is necessary for accurate detection of attacks.

#### Protocol parsing specifications

Protocol specifications (Network Security Platform's protocol parsing mechanisms) parse through network flows to validate traffic and divide it into protocol fields which may then be actively tested against McAfee-supplied attacks or Custom Attacks. By dividing protocol traffic into the appropriate fields, a Sensor can perform matches against the most specific field or subfield pertinent to an effective attack, thus resulting in very low false-positive rates. Since the parsing process is fully stateful, it allows detection of anomalies in the protocol's behavior. Additionally, this parsing makes it possible to provide an additional benefit to McAfee Custom Attack writers in the form of qualifiers. Qualifiers are tests that are embodied in the name of a particular protocol field. For example, rather than specifying that an HTTP request method must be "GET", the Network Security Platform system allows you to use "http-get-req-uri" as the name of the field, saving you the requirement of providing that test in the Custom Attack, and the Sensor from having to perform an extra pattern match.

#### Packet searches

Traffic flows that are not identified as belonging to any particular protocol are passed to the packet search protocol specification engine for further parsing. Network Security Platform presents each direction of the flow to McAfee-supplied attacks and to any Custom Attacks. Tests against packet search traffic typically take the form of specific ordered pattern matches so as to prevent false positives and performance problems.
**Attack definitions**

Attack definitions tie together elements of the above-described framework to derive specific "fingerprints" for network traffic from smaller building blocks.

In essence, attack definitions are like DNA tests. They can identify both specific people and relatives of that person. In the IPS case, the relatives may be a collection of buffer overflow attacks against a certain piece of software, and the particular person would be a specific piece of exploit code.

While the two are not greatly different, Network Security Platform adopts a convention of differentiating between attacks based on abnormality and attacks based on specific traffic. The main difference is that while anomaly-based attacks examine the network for unexpected or non-conforming behavior, specific attacks will often look for a very particular indicator, such as a flag with a particular value, or a specific string's presence. Anomaly attacks know what to expect in normal traffic, and trigger when they get something else. Normal attack definitions look for specific misbehavior. The custom attacks that you define must check for behavioral anomalies as well as specific exploit strings. Thus, all possible attempts to exploit a vulnerability can be detected.

---

**Technical information references**

A Custom Attack definition is generally based on an advisory or some other description of a known vulnerability. You should have in hand whatever information you can find regarding the attack definition. This can include traffic dumps of an attack in progress or the exploit code itself. You can use this information to determine the characteristics of the vulnerability.

You should know the specific criteria that the attack definition should comprise of, such as field values and patterns to match. Your research may lead to a long list of characteristics specific to the exploit traffic. Bear in mind, however that an attack definition based on all suspicious characteristics may be too specific. Although it would be precise, it may impact a Sensor's throughput or lead to detection problems. On the other hand, an attack definition based on only one of the characteristics may be too broad and generate false positives.

---

**Importance of testing custom attack definitions**

It is imperative that you thoroughly test your attack definitions before deploying them in a production environment. Incorrect definitions can lead to false positives, false negatives, and performance problems, any of which could be very detrimental to the security and reliability of your network. The best way to avoid these problems is to make use of a comprehensive test plan that puts the attack definition through the full deployment process and verifies that it performs as expected.

McAfee recommends that at a minimum you include the following verification exercises in your test plan:

- Use traffic generation tools or packet dumps to verify that your attack definitions match the traffic they are supposed to detect.
- If possible, verify that any custom attack definition is not duplicating functionality already available in Network Security Platform. For example, check if there is McAfee-supplied attack definition for the same condition. You can do this by examining whether your test traffic raises duplicate alerts - one for McAfee-supplied attack and another for custom attack.
- Deploy the custom attacks on a non-production Sensor connected to either a test network that mirrors your production network traffic or a non-production Sensor connected to your production network in SPAN or Tap mode.
Quick tour of the custom attack editor

To create and import Custom Attacks, you use the Custom Attack Editor - a powerful tool available in the Manager. Using this tool, you can define both McAfee Custom Attacks and Snort Custom Attacks. This section explains the user interfaces and features of the Custom Attack Editor.

Contents
- Basics of the custom attack editor interface
- Default page of the Custom Attack Editor
- Attack creation interfaces
- Test compile custom attacks

Basics of the custom attack editor interface

You use the Custom Attack Editor to manage custom attacks. You can launch it from the Custom Attacks page of the Manager. From the Manager, select Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks.

The Custom Attacks menu option is available only for the root admin domain.

The Custom Attacks page lists the following:

- Attacks: the total number of custom attack definitions currently in the Manager. This is regardless of whether a particular attack is included or not in the Network Security Platform policies.
- Signatures: the total number of constituent signatures or rules in the custom attacks that are currently in the Manager.
- Protocols: When writing the signatures for a McAfee Custom Attack, you can select the packet grep protocol. You can choose from the list of McAfee-defined packet grep protocols or create your own. This field reflects the number of such custom-defined packet grep protocols currently in the Manager. Note that selecting a packet grep protocol does not apply to Snort Custom Attack definitions.
- Last Modified: time when custom attacks were last modified in database. This is as per the system time of the server.

Click Custom Attack Editor to launch it. When you subsequently close the Custom Attack Editor, and if there are any unsaved changes in the custom attack definitions, a warning message informing that all the unsaved changes will be permanently lost is displayed. You can click Yes to discard the changes. You can click No to cancel the operation and save the changes before you close the Custom Attack Editor.
Default page of the Custom Attack Editor

This section explains the default page that displays when you launch the Custom Attack Editor.

The default page of the Custom Attack Editor has the following areas:

- **Menu Options**: The main actions that you can perform in the Custom Attack Editor.
- **Tabs**: All the custom attacks are displayed in tabbed regions in the Custom Attack Editor. All Custom Attacks is the default tab that is displayed when you launch the Custom Attack Editor. It lists all the custom attacks that are currently stored in the Manager server as well as newly created custom attacks.

You can hide or move around a column. You can also rename a column heading. You can also modify the way the content is aligned in each of the columns.

- **Viewing options**: This area contains the options you can use to list just the custom attacks that you wish to view. These options can be helpful when you are trying to locate specific custom attacks from a large set.
- **Status Bar**: When you save the changes made to the custom attacks, the Status Bar displays the progress of the action.
Menu items
This section describes the menu options that are available in Custom Attack Editor.

![Figure 3-2 File menu](image)

- **File**
  - **Save** — Save the custom attack definitions from your client to the Manager server. Note the following:
    - When you click **Save**, it applies to the attacks displayed in the current tab as well as the other tabs. That is, changes to any of the custom attacks displayed in the All Custom Attacks tab are saved in the Manager.
    - A saved attack is included or excluded in the policies of Network Security Platform based on its State. If you want to include an excluded attack, then right-click on it and change the attack's state to included and then click **Save**. You can also exclude an attack that has been included.
    - After saving the changes, you need to do push the configuration change to the Sensor for the changes to have effect on the traffic.
  - **Cancel Changes** — Click **Cancel Changes** to nullify any changes you did using the Editor and revert to the last saved version of all the attacks listed in All Custom Attacks tab.
  - **Import** — Select this menu to import Custom Attacks from files into the Manager.
    - **Custom McAfee Attack** — Select this to import previously exported McAfee Custom Attacks.
    - **Snort Rules** — Select this to import Snort Custom Attacks. You can import rules from a .config, .conf, or .rules file.
  - **Export** — Select this option to export the custom attacks in the Manager to a file.
    - **Custom McAfee Attacks** — This option exports all the custom attacks (McAfee as well as Snort Custom Attacks) and custom-defined packet grep protocols to a zip file.
  - **Snort Advanced**
    - **View Snort Variables** — You can use variables, classification types, and references when defining Snort Custom Attacks. However, these must be available in the Manager for you to use them in the Snort Custom Attacks. Use this feature to view the currently available variables, classtypes, and references in the Manager.
• Preferences
  • **Include Duplicate SNORT Attacks** — This includes the SNORT attacks that are duplicated to be listed in the *All Custom Attacks* tab.

• **Close** — Click to close the Custom Attack Editor.

---

**Figure 3-3 Attack menu**

• **Attack** — The Attack menu provides options to manage Custom Attack definitions. Note that the Attack menu options are also available when you right-click on an attack in the custom attacks display pane.

• **New** — Click this to create a Custom Attack Definition
  • **McAfee Attack Definition** — Click to define a McAfee Custom Attack. You can either create an exploit attack or a reconnaissance attack. You need to create an attack before adding signatures to it.
  • **Snort Rule** — Click to define a Snort Custom Attack.

• **Edit** — Modify the selected custom attack.

• **Delete** — Delete the selected attack from the Manager client. To delete it from the Manager server, you need to select *File | Save* after you delete it from the client.

• **Set Target Device Type** — You specify the device family to which an attack definition should be applied to. For example, if you want only the I-series Sensors to check for a specific signature of an attack, then you can choose **I-series only** as the Set Target Device Type. Then, even when you apply a policy containing the signature to an M-series Sensor, it does not check the traffic for this attack.

> When you change the **Set Target Device Type** for a McAfee Custom Attack, the same is applied to all the constituent signatures.

• **I-Series only** — Select this if you want only the I-series Sensors to check for this attack.

• **M-Series, NS-Series and VM** — Select this if you want the M-series Sensors and the NS-Series Sensors to check for this attack.

• **I-Series, M-Series, NS-Series and VM** — Select this if the attack applies to I-series, M-series, and NS-series Sensors.

You can save up to 4500 McAfee Custom Attacks in the Manager. Regarding Snort Custom Attacks, you can save up to 2500 attacks with Target Device Type set to **I-series only**; 5000 attacks with Target Device Type set to **M-series and NS-series only**. Factor these in when you select the Target Device Type. For example, consider that you have 2500 **I-series only** Snort Custom Attacks and 2500 **M-series and NS-series only** attacks in the Manager. If you assign **I-series, M-series and NS-series** as the Target Device Type for a new Snort Custom Attack, the Manager displays an error message because this exceeds the allowed number for **I-series**.
• **Include** — Select this to include an attack in the policies of Network Security Platform.

• **Exclude** — Select this to exclude an attack from the policies of Network Security Platform.

• **Test Compile the Attacks** —


• **Packet Grep**

• **Manage Packet Grep Protocols** — This option opens the Packet Grep Protocol interface where you can view the custom as well as McAfee-defined packet grep protocols. To define another custom packet grep protocol, specify the transport protocol (TCP or UDP), direction, and the destination ports. You can also delete an existing custom-defined packet grep protocol from the Manager.

Before you delete a custom-defined protocol, make sure it is not used in any of the McAfee Custom Attacks.

**Tab regions**

To enable easy viewing, Custom Attacks are listed in tabbed regions in the Custom Attack Editor. The All Custom Attacks tab is the default tab that is displayed when you launch the Custom Attack Editor. It displays all the custom attacks that are currently in the Manager server as well as the Manager client.

In addition, tabs are created when you use Display Filters to locate specific custom attack definitions. Tabs are also created when you import custom attacks from a file. Consider that you have icmp Snort rules in a file and ftp Snort rules in another. When you import the icmp rules, a tabbed region is automatically created that displays the imported rules. Now, if you import the ftp rules, one more tab is created to display the imported ftp rules. This is to help you manage custom attacks better. However, when you create a custom attack directly in the editor from a tab, it is only listed in the All Custom Attacks tab and not in the current one.

You can click on a column heading and drag-and-drop it to suit your viewing preference. You can right-click on a column heading to:

• Hide the column

• Display any hidden columns

• Rename the column heading

• Modify the alignment of the content in the column

What you see in the figure below are the default column headings. To customize a column heading name, right-click on it and select Rename.

![Figure 3-4 Tab regions](image)
Table 3-1   Tab regions

<table>
<thead>
<tr>
<th>Tab name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Custom Attacks</td>
<td>This is the default All Custom Attacks tab that lists all the custom attacks in the Manager server as well as client.</td>
</tr>
<tr>
<td>NSP Attack ID</td>
<td>The unique ID assigned by the Manager to each custom attack. The Manager assigns the same temporary ID to all custom attacks that are yet to be saved in the Manager server. When you save the attack in the database, the Manager assigns a unique ID to each attack. For McAfee Custom Attacks, this ID starts with 0xc and for Snort Custom attacks, it starts with 0xe. For McAfee Custom Attacks synchronized from the Central Manager, this ID starts with 0xcc and for Snort Custom attacks, it starts with 0xee. You can sort the listed attacks based on multiple columns by pressing the Ctrl key and clicking on a column heading. The number next to a column heading denotes the order in which the rules are sorted.</td>
</tr>
<tr>
<td>SID</td>
<td>Snort rule ID (SID) is the ID assigned to a Snort Custom Attack by you or the party that provided the rule. The Snort attack definitions that you want to save in the Manager database must have a unique SID. Make sure that the SIDs of the attacks that you are writing or importing have not been used by the definitions that are already in the database. For attack definitions that failed to import, the Manager assigns -1 as the SID. SIDs are not applicable to McAfee Custom Attacks.</td>
</tr>
<tr>
<td>Imported Snort Rule</td>
<td>This is column that appears when you import Snort rules. If you import McAfee Custom Attacks, then a column named Imported Custom McAfee Attacks is created.</td>
</tr>
<tr>
<td>Last Modified</td>
<td>This is the time stamp when a custom attack was imported or modified. This is as per the Manager client system time.</td>
</tr>
<tr>
<td>State</td>
<td>This column indicates whether a custom attack is included in the Network Security Platform policies. You can right-click on an attack and then include or exclude it. In case of a Snort Custom Attack, the Manager checks if there is an equivalent McAfee attack for the same CVE ID. If it exists, the Snort Custom Attack is imported but excluded in the policies by default. Then if you include such an attack and also enable alerting for the corresponding McAfee signature, then two alerts may be raised for the same attack traffic - one triggered by the Snort Custom Attack and the other by the McAfee attack.</td>
</tr>
<tr>
<td>Name</td>
<td>The name for the attack. In case of Snort Custom Attack, the Manager assigns this name. The format that the Manager uses for the name is SNORT:&lt;the text specified for the msg rule option in the rule&gt;(&lt;SID&gt;). So this name is modified accordingly if you modify the msg text or the SID of the rule. Note that msg rule option and a unique SID are mandatory for a Snort Custom Attack. In case of McAfee Custom Attack, you define the name. The Manager appends &quot;UDS&quot; at the beginning of the name. You can change the name that you defined.</td>
</tr>
<tr>
<td>Severity</td>
<td>An attack definition can have a severity of low, medium, or high. In case of McAfee Custom Attacks, you can specify the severity. You can also modify the severity of an existing custom attack. In case of Snort, it is based on the priority value of the rule. For a rule, this priority could be derived on its classtype or the priority option.</td>
</tr>
<tr>
<td>Format</td>
<td>This column indicates whether it is a Snort Custom Attack or McAfee Custom Attack.</td>
</tr>
</tbody>
</table>
Table 3-1  Tab regions (continued)

<table>
<thead>
<tr>
<th>Tab name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversion Result</td>
<td>This column indicates whether a Snort rule was converted to McAfee format. Some rules may have been converted but with warnings. For example, rules for which an equivalent McAfee signature exists are converted with warnings. Some rules may have failed to convert. For example, rules that use undefined variables fail to convert. This column is not applicable to McAfee Custom Attacks.</td>
</tr>
<tr>
<td>Conversion Notes</td>
<td>This column displays any error or warning messages related to the conversion of a Snort rule into McAfee's format. It may also display the metadata of a rule, if available. This is not applicable to McAfee Custom Attacks.</td>
</tr>
<tr>
<td>UDP Blocking</td>
<td>This column indicates whether the Sensor would drop just the packet that matches the custom attack or the entire flow. In case of Snort Custom Attacks, it is always set to attack-packet.</td>
</tr>
<tr>
<td>Target Device Type</td>
<td>Specify the Sensor series to which the constituent signatures or rule should be applied to. For example, if you specify M-series only, then only the M-series Sensors check for the corresponding signatures or rule.</td>
</tr>
<tr>
<td>Protection Category</td>
<td>Indicates the Protection Category to which the McAfee Custom Attack belongs. You specify this when you create a McAfee Custom Attack. This column is not applicable to Snort Custom Attacks.</td>
</tr>
</tbody>
</table>

View options

Various options are available in the tabs of the Custom Attack Editor to enable you to easily locate the information that you require. The options are:

- Detail view
- Sorting based on column values
- Group By
- Display Filter

Detail View: This is the default view that displays all the details of the listed Custom Attacks.

Sorting based on column values: You can sort the list of Custom Attacks based on the values of one or more columns. This can be in the ascending or descending order for the selected columns.

To sort the display information based on a column, click on the column heading. To base it on multiple columns, hold the Ctrl key and click on the required column headings (in the order that you want).

Group By: This option enables you to view the aggregate attack count based on certain values. For example, out of the attacks listed in a tab view, you can view the number of high, medium, and low-severity attacks.
To use the group by option:

1. Select Group by in a tabbed region.

2. Select one of the following:
   - State
   - Severity
   - Protection Category
   - Format
   - Conversion Result
   - UDP Blocking
   - Target Device Type

**Display Filter**

To locate custom attacks based on values such as severity and conversion result, you can just sort the listed attack definitions based on these values by clicking on the corresponding column heading. For advanced search capabilities, you can use the Display Filter feature. You can create a display filter with a set of criteria to view only those attack definitions that meet the criteria. For example, you can locate attack definitions based on what is contained in the attack names. This way you can easily locate specific attack definitions from a larger set.

> A Display Filter that you define is always applied on the attacks listed in the All Custom Attacks tab and not just on the attacks in the currently selected tab.

**Create a display filter**

To understand how to create a display filter, consider that you want to view only those attack definitions that meet the following criteria:

- Name contains one of the following strings:
  - Buffer-overflow
  - FTP CWD
- State is Included
- Severity is High

To create a display filter for the above criteria:

**Task**

1. In the All Custom Attacks tab of the Custom Attack Editor, click Display Filter and then click New.

2. In the Display Filter window, enter a display filter name for your reference.

3. Under Filter Criteria, click next to attack name.

4. On the right-hand side pane, select Contains from the drop-down menu and enter buffer-overflow and then click the adjacent +.

   A second row of options is displayed.

5. In the second row, click on the toggle button next to the drop-down menu until it turns to or

6. Select Contains from the drop-down menu and enter FTP CWD.

7. In the Filter Criteria pane, click that is adjacent to State.
In the right pane, under **State**, select **Equals** and **Included** from the drop-down menus.

In the **Filter Criteria** pane, click the option that is adjacent to **Severity**.

On the right pane, under **Severity**, select **Equals** and **High** from the drop-down menus. Now, the **Display Filter** window should have the criteria as shown in the figure below.

![Display Filter](image)

**Figure 3-6 Display Filter**

Click **Apply Once**.

The attack definitions that meet the criteria set are displayed in a new tab.

---

**Attack creation interfaces**

As discussed earlier, there are two types of custom attack definitions: McAfee Custom Attack and Snort Custom Attack. The methods to create the two types of attacks are different. So, the interfaces where you define them also differ accordingly. This section describes these interfaces in detail.

**McAfee custom attack creation interfaces**

To access the McAfee Custom Attack creation interface:

1. In the Custom Attack Editor, go to **Attack** | **New** | **McAfee Attack Definition**.

The McAfee Custom Attack creation interface presents information related to naming, describing, and categorizing the attack. You can either create an Exploit Attack or a Reconnaissance Attack.

**See also**

*Create custom attacks on page 41*
General section

The General section enables you to name your attack and type a description.

- **NSP Attack ID** — The numeric ID assigned for the attack by the Manager for database archival. The Manager assigns the ID after you save it in the Manager server. For McAfee Custom Attacks, the IDs begin with 0xc. For the McAfee Custom Attacks created in the Central Manager, the IDs begin with 0xcc.

- **Name** — The name you assign to the attack. "UDS" is automatically appended to the front of every created attack name when you save it in the Manager server. For example, if you name the new attack "HTTP Attack XYZ", it appears as "UDS-HTTP Attack XYZ" in the Custom Attack Editor, as well as in the attack database when you subsequently save the attack in the Manager server.

- **Description** — Use this area for notes and other pertinent information.
  
  We recommend that you type specific, useful information in the Description field for easy future reference.

- **Severity** — Select a severity from the drop-down list. Choices are High (most severe), Medium, and Low (least severe).

- **Protection Category** — You must choose a Protection Category from the available options. The Protection Category indicates the intent of the attack and the intended target. For example, you can choose Client Protection/Operating Systems for an attack targeting vulnerabilities in client operating systems. In this example, Client Protection is the category and Operating Systems is a subcategory. The list of Protection Categories is pre-defined and is provided by McAfee Labs. You cannot modify it. This list is updated when you update the Signature Set.

- **Target Device Type** — You can choose to apply a McAfee Custom Attack signature for just the I-series, M-series, NS-series and VM, or for all of them. The value for this field depends on what you select for the constituent signatures. For example, if the attack contains signatures with Target Device Type set only to I-series, then the value displayed here is I-series. If some signatures are set to M-series, NS-series and VM, and some to I-series, then the value displayed here is I-series, M-series, NS-series and VM. You cannot edit this field at the attack level, but the Manager modifies it accordingly when you change it for the corresponding signatures.
  
  To set the same Target Device Type to all the signatures of an attack in one go, right-click on a custom attack and select the required value for Set Target Device Type.

- **Blocking Logic** — You can specify whether the Sensor should drop just the packet that matches the custom attack or the entire flow.

- **Direction** — You can choose the direction of the attack to be blocked. It could be either from the client to server, or server to client, or follow the attack packet.

Matching Criteria section

The Matching Criteria section enables you to categorize your attack for eventual submission to your Manager's attack database. Once exported to the attack database, selection of your attack is included by one of several rule sets, which are then added to policies for enforcement.

A rule set is defined as a set of ordered rules used to determine what attacks or conditions are of interest, and thus should be monitored when applied as part of a policy. A rule set is configured based on attack category (Exploit, Reconnaissance), operating system (Windows, UNIX), protocol (HTTP, DNS), application (SendMail, Apache), severity (High, Low), and benign trigger probability (High, Low) options. Except for benign trigger probability, you need to indicate these options when defining at the
attack level. You can indicate the value for benign trigger probability when creating the signatures for an attack definition. Based on the values that you set for these options, the attack is available for policy enforcement.

- **Software Package** — Manager assigns tcpip-machine to all Snort Custom Attacks. You cannot edit this value.

- **Protocol Only** — This is the identified impact protocol for the Snort Custom Attack.

**Signature creation interface**

After you define the details for a McAfee Custom Attack, you need to define one or more signatures for that attack. To access the signature creation interface:

- Click **New** in the **Signature** section of the **Add Exploit Attack** window.

The signature creation interface presents the details related to the signature you are creating. These details are used to provide further information about the suspicious activity for which you are attempting to capture and analyze through Custom Attacks. By configuring each field, you are further refining the signature's search, which raises the probability your attack will successfully detect the desired activity, thus preventing false positives. The signature detail fields are as follow:

- **Signature ID** — The Manager assigns this value when you save the signature. You cannot edit this field.

- **Signature Name** — The name you give the signature. If the attack is detected, the Threat Analyzer displays the detecting signature in the attack's details window.

- **Benign Trigger Probability** — This is an indication of the probability of that your signature will alert on traffic that may not be an attack. The choices are High, Medium, and Low, with Low representing approximately a 0% to 33% chance of your signature raising a false positive. For example, your signature may be a generic string search, such as "Confidential", thus the BTP would be best graded as High. If you know a specific file name that is infected or sensitive, such as "program.exe", then you could set your BTP to Low to reflect your confidence in your signature.

**Figure 3-7  New signature window**

- **Target Host Architecture** — You can define a specific machine architecture targeted by an attack with shellcode. For example, to detect a Intel-specific shellcode, you can select i386 as the architecture. The default is any.

- **Detection Window** — This field describes where in a flow you want your signature to actively check traffic. The choices are as follows:
  - **packet** — check each packet for your signature condition(s). If your condition is xyz, and the Sensor sees xy in one packet and z in the next, no alert is generated. Each condition you specify must be found in one packet.
  - **request** — check only the request direction of a flow.
• response — check only the response direction of a flow.

• flow — check the entire flow (between two hosts) for your condition(s). If your condition is xyz, and the Sensor sees xy in one packet and z in the next of the same flow, an alert is generated.

• Target Device Type — You can choose to apply a McAfee Custom Attack signature for just the I-series, M-series, NS-series and VM, or all of them. The value for this field depends on what you select for the constituent signatures. For example, if the attack contains signatures with Target Device Type set only to I-series, then the value displayed here is I-series. If some signatures are set to M-series, NS-series and VM, and some to I-series, then the value displayed here is I-series, M-series, NS-series and VM. You cannot edit this field at the attack level, but the Manager modifies it accordingly when you change it for the corresponding signatures.

To set the same Target Device Type to all the signatures of an attack in one go, right-click on a custom attack and select the required value for Set Target Device Type.

• Conditions — A condition is the test or group of tests that, if met, raises an alert. Conditions are made up of protocol field tests, or comparisons. A signature may have multiple conditions, each with multiple comparison tests. If you configure multiple conditions for a signature, each condition must be met in order (AND THEN logic) before an alert is raised.

• ADD — Add a new condition.

• DEL — Delete a selected condition.

• Comparisons — Within a condition, you add one or more comparison tests. Comparisons can be AND or OR in nature. AND comparisons must be met for an alert to be generated for malicious traffic. OR comparisons allow for multiple comparisons within a condition, of which only one of the OR tests must be met to raise an alert. You must start a condition with an AND comparison. You may have up to 32 comparisons per condition.

• AND — Add an AND comparison.

• OR — Add an OR comparison. You must have already added an AND comparison.

• DEL — Delete a selected condition.

• General
  • Expand All — maximizes all conditions for viewing comparison details.

See also
Create a signature on page 44
Create the signature for the example on page 149
Create the signature on page 142

Packet search protocol interface
Not all traffic can be supported by standardized protocol specifications. There are cases that require a different means of identifying an attack based on packet searching. McAfee has a pre-defined list of packet grep protocols. You can view this list of protocols in the Packet Grep interface. Using this interface, you can also create packet grep protocol in case the one that you need are not in the list of McAfee defined packet grep protocols.

To access the Packet search protocol interface:

• In the Custom Attack Editor, select Packet Grep | Manage Packet Grep Protocols.

McAfee default protocols
The McAfee Default Protocols section lists various applications that are commonly used/allowed, but that do not use a standardized protocol, and which may also be used for malicious purposes. The Custom Attack Editor enables you to monitor usage of programs such as NetMeeting and PCAnywhere.
in order to prevent external attacks. For example, your security policy may allow the use of PCAnywhere to resolve desktop/server issues remotely, but the program can also be used to infiltrate your network and perform malicious acts. The Custom Attack Editor enables you to create a pattern-matching signature for responses or requests to/from any of the listed programs for monitoring purposes.

To create a signature for any of these instances, simply create a signature instance, and select Packet Grep Protocol as a comparison for a condition.

**Custom-defined packet grep protocols**

Custom-defined packet grep protocols section is where you can view the list of custom-defined packet grep protocols and also create new ones.

- **Add** — Add a new packet grep protocol instance.
- **Del** — Delete a created instance.
- **View** — View a created instance.

You cannot modify an existing custom-defined packet grep protocol.

![Custom-defined Packet Grep Protocols window](image)

**Figure 3-8** Custom-defined Packet Grep Protocols window
When you add a packet grep protocol, you need to specify the transport layer protocol, the process method, and the identifying port(s) for identifying those attacks that use an application protocol unknown to Network Security Platform.

![Packet Grep Protocol](image)

**Figure 3-9 Add a new packet grep protocol**

The interface fields and options are as follow:

- **Name** — abbreviated name displayed during selection. For example, you would type "luckyp" here, then type "Lucky Protocol" for the Verbose Name.

- **ID** — numerical ID assigned by the system for database archival.

- **Name** — full name for custom protocol. For example, if you typed "CVeritasBackup" at Name, you would type "Custom-defined Veritas Backup" here.

- **Transport Protocol** — Specify the transport protocol that will be used by the application whose protocol details that you are defining.
  - TCP
  - UDP

- **Direction to Process** — transmission direction wherein you want to search for a specific activity.
  - Process Request
  - Process Response

- **Ports** — target ports that have been added for the current custom protocol.
  - **Port Number** — type the number in this field then click Add to add a port that can be used to identify the unknown traffic. Click Delete to delete a selected port number.

  Do not enter any standard port number.
Snort custom attack interfaces
You create Snort Custom Attacks using the Snort Rules Language. When you write a Snort rule in the Custom Attack Editor and save it in the Manager server, an attack definition is automatically created and the rule associated with this attack.

Interface to write Snort rules
The interface for writing a Snort rule is a free-text editor within the Custom Attack Editor. To access the Add Snort Rule editor:

- In the Custom Attack Editor, select Attack | New | Snort Rules.

Enter the Snort rule in the Add Snort Rule editor and click Save. The rule and the associated Custom Attack are saved in the Manager client. You can view this attack in the currently selected tab as well as in the All Custom Attacks tab.

Interface to edit snort custom attacks
To access the Edit Snort Attack interface, right-click a Snort Custom Attack and select Edit.

Figure 3-10  Edit Snort Attack window
When you create a Snort rule in the Manager, based on the elements of the rule, the Manager assigns values to some of the fields. If you modify the rule in the Raw Snort Rule Text section, the Manager modifies the field values accordingly. For example, if you change the msg in the rule, the attack and signature names also change because the Manager assigns the attack and rule names as per the msg keyword in the rule.

Every time you modify the rule in the Raw Snort Rule Text section, you need to check if the rule is conforming to Snort rules syntax by clicking Validate. After the Manager translates the rule to McAfee's format, click Save to save the changes in the Manager client. The changes are saved in the Manager server only when you click Save in the File menu of the Custom Attack Editor.

The following the fields displayed in the General tab for a Snort Custom Attack:

- **NSP Attack ID** — The numeric ID assigned for the attack by the Manager for database archival. The Manager assigns the ID after you save it in the Manager server. For Snort Custom Attacks, the IDs begin with 0xe. For Snort Custom Attacks created in the Central Manager, the IDs begin with 0xee.

- **Name** — The name that the Manager assigns based on whether you created it from the Central Manager or Manager, the msg option, and the SID of the rule. To modify the attack name, you can modify the msg value or the SID, and save the Snort Custom Attack to the database.

- **Description** — Description for the Snort Attack.

- **Severity** — Severity of the Snort Attack. It can be high, medium or low.

- **Protection Category** — Displays the Protection Category assigned to the attack. The Protection Category indicates the intent of the attack and the intended target. The list of Protection Categories is pre-defined and is provided by McAfee Labs. You cannot modify it. This list is updated when you update the Signature Set.

- **Target Device Type** — You can apply a Snort Custom Attack signature for just the I-series, M-series and NS-series, or for all of them. The value for this field depends on what you select for the corresponding rule. You cannot edit this field at the attack level, but the Manager modifies it accordingly when you change it for the corresponding rule.

  By default, you can apply up to 2500 Snort Custom Attacks per I-series Sensor, and up to 5000 per M-series Sensor and NS-series Sensor.

- **Blocking Logic (UDP attacks only)** — In case of Snort Custom Attacks, the Sensor drops just the packet that matches the rule. You cannot edit this field.

- **Direction** — Displays the direction assigned for parsing the attack traffic.

The following fields are displayed in the Matching Criteria tab for a Snort Custom Attack:

- **Software Package ID** — Manager assigns tcpip-machine to all Snort Custom Attacks. You cannot edit this value.

- **Protocol** — This is the identified impact protocol for the Snort Custom Attack.

- **OS** — This impacted operating system. The default value is any, which you cannot edit.

The following fields are displayed in the Signatures tab for a Snort Custom Attack:

- **Name** — This is the same as Attack Name, described above.

- **ID** — The Manager assigns this value, which you cannot edit.

Beginning with version 8.0, Snort custom attacks are translated into a newer McAfee signature format. This is required to support more Snort rule options as well as for performance improvement. However, the Snort custom attacks in this newer format are incompatible with 7.x Sensors. So, to support a heterogeneous Sensor environment, 2 signatures are created for each Snort custom attack - one for 8.0 Sensors and the other for 7.x Sensors.
After you upgrade the Manager from 7.x to 8.0 or later, it is mandatory that you re-submit all the Snort custom attacks for translation. Then, two signatures are created for those rules as well. To re-submit the rules, in the Custom Attack Editor, select File | Snort Advanced | View Snort Variables | Re-Submit Rules using Current Variables.

The two signatures are created regardless of whether you have a 7.x Sensor in your setup. This is to address scenarios where one might add a 7.x Sensor to an 8.0 Manager at a later time. If you do not require the signature for 7.x Sensors, you can turn it off.

1. Locate the ems.properties file. On the Manager server, go to `<Manager install directory>\App\config\`.

2. In the ems.properties file, un-comment `# iv.snortimport.translation.tpuverion.0.support=false`. That is, change this line to `iv.snortimport.translation.tpuverion.0.support=false`.

3. Re-start the Manager service.

- **Benign Trigger Probability** — This is an indication of the probability that the Snort Custom Attack will alert on traffic that may not be an attack. The default value is Medium, which you cannot modify.

- **Architecture** — This is the machine architecture that the attack definition is intended for. This is set to any, and you cannot modify it.

- **Detection Window** — This field indicates the section of a flow, the Sensor checks for the rule. The possible values are:
  - packet — checks each packet for the rule. If the content value of the rule is xyz, and the Sensor sees xy in one packet and z in the next, no alert is generated. Each condition you specify must be found in one packet.
  - request — checks only the request direction of a flow.
• response — checks only the response direction of a flow.
• flow — checks the entire flow (between two hosts). If the content value is xyz, and the Sensor sees xy in one packet and z in a different packet but in the same flow, an alert is generated.
• Raw Snort Rule Text — This is the only directly editable field in the General tab.
• Conversion notes — This section contains the information that the Manager logs when saving a Snort Custom Attack in its database. You can check whether the Snort rule was successfully converted to McAfee's format, if not then the probable reason to help you troubleshoot.

The Manager derives the detection window in the following order of precedence:

1. If there is no non-payload option in the rule, then the detection window is set to packet.
2. If flow option is present in the rule, then it is set to:
   • response — if the flow option is to_client or from_server
   • request — if the flow option is to_server or from_client
3. If a standard port number is the destination port, then it is set to request.
4. If the Manager is unable to derive the detection window by any of the above methods, it is set to "flow".

The Manager modifies the detection window accordingly if you modify the rule. For example, if you modify the rule by specifying a standard port for destination, then Detection Window is also modified accordingly.

• Target Device Type — See the description above in this section.
• Raw Snort Rule Text — See the description above in this section.
• Conversion Notes — See the description above in this section.

Test compile custom attacks

The Snort rules that you create or import must be compatible with Network Security Platform. When you save the custom attacks in the Custom Attack Editor, the Manager compiles them to check their compatibility. If there are any incompatible custom attacks, a critical fault message named Incompatible custom attack is raised. Information regarding the reason for the compilation failure is provided in the fault message. You must correct the incompatible custom attacks and then save them again. If you do not correct them, the subsequent signature set update to the Sensors might fail.

After you correct the incompatible custom attacks, you can use the test compile feature in the Custom Attack Editor to verify the rules for compatibility even before you save them in the Custom Attack Editor.

Task
1. In the Custom Attack Editor, select the custom attacks that you want to test compile.
2. Select Attack | Test Compile the Attacks.

You can access this option from the right-click menu as well. A success or failure message is displayed after the test.
Mechanics of a custom attack

This section explains the elements that make up a McAfee Custom Attack. A good understanding of these elements and how they can impact your IPS setup is required to create effective attack definitions.

See also
Create custom attacks on page 41

Contents
- Structure of a custom attack
- Signature test reference
- Performance issues
- Considerations

Structure of a custom attack

A McAfee Custom Attack is like a top-level container in which you define one or more signatures. Some of the critical values that you need to specify when defining an attack are the attack severity, the impact package, and protocol. Based on these values, the Manager categorizes the attack and includes it in the policies.

To save a custom attack in the Manager server you must specify at least one impact package or protocol.

Structure of a signature

Network Security Platform’s signature support is very comprehensive. In addition to the normal pattern matching and numeric tests supported by many network IPS systems, Network Security Platform allows signatures to have a complex structure that can order and group the tests necessary for accurate detection.
How tests are logically structured
Network Security Platform provides a simple but powerful set of logical constructs that will allow you to create an attack definition to match the fingerprint of nearly any network activity. The logical constructs provide methods for grouping and ordering the more granular pieces of the signature - numeric tests and pattern matches - in any order you require. The core constructs are:

- **AND** - Tests are grouped within AND blocks. The tests must all succeed, but may occur in any order inside the context of a given flow. AND blocks cannot be contained within an OR condition.

- **OR** - The first test to succeed fulfills the condition, and the next AND or OR block is engaged. OR constructs are contained within AND blocks, though at times they may be the only contents.

- **ANDTHEN** - ANDTHEN is the ordering construct. By placing ANDTHEN statements between AND blocks, you can create a structured order for your tests, if required.

The actual tests of which signatures are composed are placed inside either AND or OR constructs. These are called conditions and are like containers. ANDTHEN is used to link the AND containers (conditions) together to form a larger structure.

Supported test methods
A McAfee Custom Attack supports 2 basic classes of test:

- **Pattern Match** - Pattern matches can be used to match ASCII or binary strings. Support for common regular expression constructs is available.

- **Numeric Comparisons** - Numeric comparisons can be used to test for a match with a given value or a range of values, depending on the numeric comparison test chosen.

Other signature components
In addition to the mechanics of the signature itself, Network Security Platform requires a few other pieces of information to do its job properly:

- **Detection Window** - The detection window is the portion of the traffic flow that the Sensor should check against an attack definition for matches. The detection window may be one of:
  - **Packet** - All of the signature tests (and thus all protocol fields) must occur in a single packet on the network.
  - **Request** - All of the signature tests must occur in the request direction of the flow. (Useful if the protocol is bidirectional and might include the same data or commands in either direction.)
  - **Response** - Identical to request, but used to examine only response-direction traffic.
  - **Flow** - When the detection window is set to flow, all conditions can be fulfilled by traffic flowing in either direction. Tests are still subject to any ordering imposed by the structure of the signature or flow direction implied by the field being tested.

  It is possible to create a correct signature that will never trigger if the detection window is set incorrectly. This should be one of the first items you check when troubleshooting a custom attack definition.

- **Benign Trigger Probability** - This is a measure of the confidence with which you believe the signature can identify a network event. A high benign trigger probability (BTP) implies that a signature may be prone to false positives. When creating the signature, you should set the BTP to "Low" to ensure that your signatures are selected by the Default Policy.

Create more than one signature per attack
The capability of creating more than one signature in an attack definition is part of what allows Network Security Platform to keep its false positive rate so low. You should create multiple signatures for an attack, when possible, so as to be both more inclusive and more specific. You may want to
create a generic signature that will catch all attack variants (including those which are unknown to you). You should also create more specific signatures to detect particular pieces of exploit code, if possible. This can be valuable in providing pointers on what to investigate if a machine has been compromised, in addition to keeping your rate of false positives far below that of the average signature-based IPS.

Signature test reference

This section describes each test type available for use in signatures and the required and optional parameters for each.

See also
Create a signature on page 44

Numeric comparison tests

- `!=`: Test of numeric inequality
- `==`: Test of numeric equality.
- `unsigned <`: True if the unsigned Value specified is less than the value obtained from the network traffic.
- `signed <`: True if the signed Value specified is less than the value obtained from the network traffic.
- `unsigned >`: True if the unsigned Value specified is greater than the value obtained from the network traffic.
- `signed >`: True if the signed Value specified is greater than the value obtained from the network traffic.
- `unsigned <=`: True if the unsigned Value specified is less than or equal to the value obtained from the network traffic.
- `signed <=`: True if the signed Value specified is less than or equal to the value obtained from the network traffic.
- `unsigned >=`: True if the unsigned Value specified is greater than or equal to the value obtained from the network traffic.
- `signed >=`: True if the signed Value specified is greater than or equal to the value obtained from the network traffic.

Numeric test parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Field</td>
<td>The field that will be compared with the supplied value.</td>
</tr>
<tr>
<td>Value</td>
<td>The value that should be compared against the value retrieved from the network traffic. This field does not support floating point values.</td>
</tr>
</tbody>
</table>
Numeric range tests

- **signed-in-range** - True if the value obtained from the network traffic falls within the specified signed range.
- **unsigned-in-range** - True if the value obtained from the network traffic falls within the specified unsigned range.

McAfee recommends using an in-range test as it is more efficient than -gt and --lt tests.

**Numeric range test parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Field</td>
<td>The field that will be tested for equality with the supplied value.</td>
</tr>
<tr>
<td>Lower-bound</td>
<td>Lower boundary of the comparison range. (Inclusive)</td>
</tr>
<tr>
<td>Upper-bound</td>
<td>Upper boundary of the comparison range. (Inclusive)</td>
</tr>
<tr>
<td></td>
<td>Note that the Lower-bound and Upper-bound fields do not support floating point values.</td>
</tr>
</tbody>
</table>

Numeric enumeration tests

- **unsigned ==** - True if one of the enumerated values matches the value obtained from the network traffic.

**Numeric enumeration test parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Field</td>
<td>The field that will be tested against the values derived from network traffic.</td>
</tr>
<tr>
<td>Enum values</td>
<td>Values to be compared against the values derived from network traffic.</td>
</tr>
<tr>
<td></td>
<td>Note that the values cannot be floating point or negative.</td>
</tr>
</tbody>
</table>

Pattern matching

Pattern matching within signatures is always implemented with string-match. String-match accepts a number of optional and required parameters. In the Regular Expression to Match text box, construct the string that contains the pattern to match. By default, pattern matches are case-sensitive. String-match accepts regular expressions in addition to simple strings.

**Pattern match test parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Field</td>
<td>The field that will be tested for a match with the supplied pattern.</td>
</tr>
<tr>
<td>Regular Expression to Match</td>
<td>Pattern to be matched.</td>
</tr>
</tbody>
</table>
### Create a fixed field signature

*Fixed fields* are those protocol fields that are provided by the basic IP protocols IP ("ipv4"), TCP, UDP and ICMP. They are "fixed" by contrast with the calculated nature of many provided by other protocol specifications. Due to the fact that these fixed fields occur in most packets the Sensor inspects, it is impractical for performance reasons to inspect them in the same fashion as standard protocols. For the same reason, the number of fixed field comparisons is limited to 50.

Two types of tests for fixed fields are available. Both are numeric because the fixed fields are all numeric.

- **fields-eq** - Fields-eq is a special test that is only available for fixed-field testing. It allows you to compare the values in two protocol fields. This is useful if, for example, you want to compare whether the source and destination IP addresses specified by packet headers are the same.

- **All of the numeric tests described for normal signature tests**. This includes both signed and unsigned tests.

The "Fixed Field Value Comparison" in the Custom Attack Editor also provides an "Enable BitMask" function. This is not a normal "bitmask". Instead, starting from the offset you provide (in the selected fixed field), it selects the number of bits you specify, and then uses those bits to create an integer value that is compared against your provided comparison value.

### Packet search protocols

*Packet search protocols* are a form of generic packet inspection capability used by IPS experts when dealing with protocols that are not specifically supported by the Network Security Platform system. They are generally used for implementing detection on specific ports in an efficient fashion. In these cases, it can be valuable to separate out traffic running on particular ports for processing. This can make detection more efficient because pattern matches and other tests will only examine a limited subset of network traffic. In addition, it is possible to create a packet search protocol to exclude certain traffic from being considered for detection.
Performance issues

• **False Positives** - Large numbers of false positives could (in severe cases) overwhelm the Manager. More commonly, they will simply overwhelm the user due to volume and speed of appearance.

• **Too Many Pattern Matches** - If the pattern your signature searches for occurs too often in the network traffic being analyzed, checking each match to determine whether it fulfills a signature condition may degrade Sensor performance.

  The most common cause of overwhelming numbers of pattern matches is making them too short. A pattern to be matched should never be shorter than 3 characters because the likelihood of a chance occurrence of such a pattern in normal traffic is very high.

• **Too Many Fixed-Field Tests** - If you configure too many fixed-field tests in your signatures, Sensor performance may be degraded. This is due to the fact that the Sensor will have to inspect this field in every packet of the traffic type being tested. As with pattern matches, checking each test result to see if it fulfills a signature condition may affect Sensor performance.

• **Testing Overly Common Fields** - If your signature contains a test of a field that occurs very often in the traffic type being analyzed, it is possible to degrade Sensor performance. This can be due to the number of tests that will be performed or due to the number of tests that must be checked for fulfilling a signature test condition.

Considerations

Consider the following when you create McAfee Custom Attacks. These should not present an obstacle to most signature writers.

• You can save up to **4500** McAfee Custom Attack definitions in your Manager.

• Each McAfee Custom Attack may not have more than **16** different signatures.

• Each signature may not contain more than **32** different tests, either numeric or pattern-matching.
Managing custom attacks

This section provides the information required to create and maintain McAfee Custom Attacks with one or more signatures.

Contents
- Create custom attacks
- Create an exploit attack without template
- Configure custom reconnaissance attack definition
- Import previously exported custom attacks
- Importing McAfee-defined custom attacks
- Work with packet search protocol
- Regular expression language
- Limitations of a custom reconnaissance attack

Create custom attacks

The following is the high-level recommended approach for creating McAfee Custom Attacks.

Task
1. Get started. Be clear on what you are trying to achieve through these attacks. See if there are any alternatives or better methods to achieve the same. Identify the related protocols, applications, hardware and software platforms, and so on for the attack. Make sure you have all the required information at hand.

2. Understand the mechanics of a McAfee Custom Attack. Before you get down to creating one, it is recommended that you know what goes into a McAfee. You also need to understand the probable impact of your McAfee Custom Attacks on your network.

3. Take a quick tour of the Custom Attack Editor. Get familiar with the Custom Attack Editor - the tool that you will use to manage Custom Attacks.

4. Create a McAfee Custom Attack with one or more signatures.

5. Verify if the attack is included in the policies.

6. Push the attack with its signatures to the Sensors for attack detection to happen.

See also
Creating custom attacks on page 3
Mechanics of a custom attack on page 3
McAfee custom attack creation interfaces on page 25
Viewing a policy to verify inclusion of the attack on page 132
Create an exploit attack without template

This section explains how to create McAfee Exploit Attacks and the constituent signatures.

To create a McAfee Exploit Attack instance, you start by first adding a new attack. Attacks are configured first because signatures logically relate to attacks as one particular means of detection.

Attack creation includes impact categorization for proper policy integration and enforcement. Each policy-those provided with Network Security Platform and those you create-consists of rule sets, which contain multiple categories of attacks. Attacks are categorized by the protocols, operating systems, and applications they impact. Within the Custom Attack Editor, you define the impact categories wherein your attack definition best fits, so that when you save it in the Manager, it will be included in one or more rule sets.

**Task**

   The Custom Attack Editor opens with the existing Custom Attacks listed in the All Custom Attacks tab.

2. Select Attack | New | McAfee Attack Definition | Exploit Attack.
   The Add Exploit Attack interface opens.

![Figure 5-1 Add Exploit Attack window](image)

3. Leave the default option as is and click Next.

4. In the Name field, type a new name for your attack. "UDS" (User-Defined Signature) is appended at the front automatically when you save the attack. For example, if you name the new attack "HTTP Attack XYZ", it appears as "UDS-HTTP Attack XYZ" in the Custom Attack Editor as well as in the attack database when you save the attack.

   ![The NSP Attack ID is provided by the Manager when you save it in the Manager server.](image)

5. Optionally type a Description for your attack.
   This area can be used for your notes or other specific information pertinent to your new attack.

6. Select a severity for your attack by toggling the drop-down list. Choices are High (most severe), Medium, and Low (least severe).

7. Select the most appropriate Protection Category for the attack.
8 In case of UDP attack traffic, specify whether the Sensor should block just the attack packet or the entire flow.

You set Target Device Type only in the signatures.

9 From the Direction drop-down list, select the traffic direction that must be defined in this custom attack.
   - Client to Server
   - Server to Client
   - Follow Packet

10 In the Matching Criteria tab, click Add.

   - Protocol example: If you select Low for Severity and HTTP for Exploit, the attack is applied only when a policy employs a rule set that includes Low severity and HTTP protocol attacks. The Default rule set includes HTTP attacks, but requires attacks to have a minimum severity of 2 (Low); attacks with severities of 0 or 1 (also Low) will not be included when applying the Default rule set.

   - Software Package example: If you select NetMeeting for Package, the attack is applied only when a policy employs a rule set that includes NetMeeting attacks (that is, attacks that specifically target, thus impact, NetMeeting).

You can add multiple Protocol and Package categorizations for a single attack. This enables your custom attack to be enforced by multiple rule sets and detected in any flows that meet any the selected criteria. For example, if you specify the attack to be categorized under HTTP and SMTP as well as Windows 2000 and Linux, the Sensors search all applicable HTTP, SMTP, Windows 2000, and Linux transmissions for the attack. Thus, if your attack impacts multiple protocols and/or packages, categorize the attack under each protocol and/or package.

   It is mandatory that you specify at least one impact package or protocol. If you are not sure about the impact package or protocol, then select tcpip-machine as the impact package.

   Check the rule sets (Inbound and Outbound) of the policy you plan to enforce to determine whether your attack will be selected. You also have the option of creating a new rule set in order to enforce your Custom Attack.

a Complete at least one of the following:

   1 Categorize the attack definition by protocol. For example: HTTP, FTP, DNS. In the Add Protocols & Packages dialog box, select Protocol Only and then select a protocol from the Protocol List.

   2 Categorize by software package (that is, application and operating system). For example: Internet Explorer on Windows 2003. In the Add Protocols & Packages dialog, select Package and then select one package from the list. Optionally, select an Operating System.

b Click Finish to save.

c Repeat steps to specify more protocols or packages.

11 Once your attack configuration is complete, continue to Creating a Signature.

**Tasks**

- *Create a signature on page 44*
Create a signature

Create one or more signatures to detect an attack. This section assumes that you are adding signatures to the attack that you just created in the previous section. That is, the Add McAfee Attack interface is open and the attack you created is yet to be saved in the Manager server. If you are adding signatures to a previously saved attack, double-click on the attack in a tabbed region to open the Edit McAfee Attack interface.

You can create up to 16 signatures per attack definition.

**Task**

1. In the Add McAfee Attack interface or Edit McAfee Attack interface, right-click on the attack name on the left-hand side pane and select **Add Signature**.

   The signature entry appears staggered beneath the parent attack entry. The signature-specific settings open in the right pane.

2. Optionally clear the **Signature Name** and type a new one.

3. Select a **Benign Trigger Probability** value.

4. Select a **Target Host Architecture**. The default is **Any**.

5. Select a **Detection Window**. The choices are as follows:
   - packet
   - request
   - response
   - flow

6. Based on the Sensor model that you plan to use for this example, select the **Target Device Type**.

7. Add one or more conditions for your signature.
To add a condition, do the following:

a  Click **ADD** under Conditions. **Condition 1** appears in the blank white field.

b  Click **Condition 1** so that it is highlighted.

c  Do one of the following:
   1  Click **AND** under Comparisons to add a single AND comparison.
   2  Click **OR** under Comparison to add multiple OR comparisons.

![Figure 5-3 Adding comparators](image)

Your selection affects what appears in the subsequent dialogs. The **String Pattern Match** option requires knowledge of the Regular Expression Language.

d  Select a comparison type from the **Comparison List** and click **Next**.

Your selection affects what appears in the subsequent dialogs. The **String Pattern Match** option requires knowledge of the Regular Expression Language.

e  Select a protocol from the **Protocol List** and click **Next**.

The subsequent options are based on the selected protocol.

- If you selected **Packet Grep Protocol** from the Comparison List, the available protocols are those packet grep instances created/provided within the **Packet Search | Manage Packet Search Protocol** feature.

f  Configure the fields for the comparison(s) you have chosen.

- If selection fields (drop-down menus) are blank, then the protocol and comparison you have chosen cannot be used to create a signature condition; select another protocol and/or comparison combination.
For example, if you want to configure a string match in the URI path of an HTTP GET request:

1. Select `req-uri-path` for the Protocol Field and select `get` as the http-req-method.

![Configure Comparison window](image)

**Figure 5-4 Configure Comparison window**

2. From the drop-down list in the Regular Expression to Match section, select the matching criteria as Equals or Does NOT Equal based on the matching criteria required for the string pattern entered.

3. If you are configuring a string pattern match, type your pattern in Regular Expression to Match.

4. Optionally, click the Case Insensitive check box if you want the pattern to be matched regardless of [letter] case.

5. Click Validate to verify that your pattern is valid.

6. Click Finish when done with the Configure Comparison fields. Your comparison appears under Condition 1.

7. Do one of the following:
   - Click ADD under Conditions to add another condition.
   - Select Condition 1 and click AND under Comparisons to add another AND comparison.
   - Select Condition 1 and click OR under Comparisons to add multiple OR comparisons to your condition. Note that the first of the OR comparison appears under (AND)(One Of).
   - (Optional) Do one of the following if you have created multiple conditions:
     - Select a condition and click DEL under Conditions to delete a condition, or select a comparison and click DEL under Comparisons to delete a comparison.

8. Optionally, you can add more signatures to your attack.
9  Click **Save** in the Add Exploit Attack or the Edit Exploit Attack interface.
   The created Attack and its signatures are saved in the Manager client that you are logged on to.

10  In the Custom Attack Editor, select **File | Save**.
    All the Attacks in the Manager client (including the one that you just created) are saved in the Manager server.

    **If you close the Custom Attack Editor without saving the attack, that is, without completing step 10 above, the attack that you created (along with any unsaved changes to other attacks) is lost.**

**See also**
- *Signature creation interface on page 27*
- *Signature test reference on page 37*
- *Regular expression language on page 70*
- *Create a packet search protocol on page 66*
- *Viewing a policy to verify inclusion of the attack on page 132*

### Templates for McAfee custom attacks

You can use the predefined templates to create some of the commonly used McAfee custom attacks. In terms of syntax and effectiveness, a McAfee custom attack is the same whether you used a template to create it or not. However, using these templates, you can create effective McAfee custom attacks even if you do not possess detailed knowledge of the related protocol, its header, or the syntax of McAfee custom attacks.

Predefined templates are available to create McAfee custom attacks that:

- Detect a URL
- Detect an email attachment file name
- Detect a DNS query or response
- Detect a string in an application running over a custom port
- Detect a TCP connection attempt from a specific IP address

When you use the templates, a McAfee custom attack with the relevant protocol is automatically created. Also, the corresponding signature is created for this attack. You can add more conditions to this signature or add more signatures to this McAfee custom attack like how you would do traditionally for McAfee custom attacks. For example, when you use the template to detect TCP connection attempts from specific IP addresses, the signature for the IP address that you specify is automatically created. To specify more IP addresses to this attack, you need to create the corresponding signatures for those IP addresses.

**By default, the signatures that are created when you use the templates are of Benign Trigger Probability (BTP) value high. You can edit this value post-creation. Note that the attacks of BTP value high are not included in the default IDS and default inline IPS policies.**

**When you save the McAfee custom attacks in the Manager database, an informational fault is displayed in the Status page to indicate whether the custom attacks were successfully saved.**

### Create a McAfee custom attack to detect a URL

You can use the pre-defined template to create a McAfee custom attack to detect a URL.
Task

1. In the resource tree, select Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks | Custom Attack Editor.
   The Custom Attack Editor opens with the existing custom attacks listed in the All Custom Attacks tab.

2. Select Attack | New | McAfee Attack Definition | Exploit Attack.

3. On the Templates page, select Detect a URL and click Next.

4. Enter a Name and Description for the attack.
   UDS (User-Defined Signature) is appended at the front automatically when you save the attack. For example, if you name the new attack HTTP Attack XYZ, it appears as UDS-HTTP Attack XYZ in the Custom Attack Editor as well as in the attack database when you save the attack.
   
   The Manager provides the NSP Attack ID when you save the attack.

5. Select the appropriate Severity for the attack.

6. Select the most appropriate Protection Category for the attack.
   • You can skip the Blocking Logic field as it is applicable only for UDP.
   • You set Target Device Type only in the signatures.

7. From the Direction drop-down list, select the traffic direction that must be defined in this custom attack.
   • Client to Server
   • Server to Client
   • Follow Packet
8  Enter the URL that is to be detected, then click Finish.

If you are specifying the protocol, you can specify only http or https in the URL. The attack is listed in the All Custom Attacks tab.

**Figure 5-5** Create a McAfee custom attack to detect a URL
9 Right-click the attack that you created in the **All Custom Attacks** tab and select **Edit**.

In the **Matching Criteria** section, the protocol is automatically selected as HTTP.

The signature for the URL that you specified is created automatically with the default values. To edit these values, select the signature and click **Edit**. You can modify the default values. You can also add more conditions to the signature. From the **Edit Exploit Attack** window, you can also add more signatures to the attack.

![Edit Exploit Attack](image)

**Figure 5-6 Details of the McAfee custom attack**

10 To save the McAfee custom attack to the database, in the **Custom Attack Editor**, select **File** | **Save**.

Until you save the attack in the database, the **NSP Attack ID** column in the **Custom Attack Editor** shows as **pending**. The Sensor detects this attack after a configuration update.

**Create a McAfee custom attack to detect an email attachment by file name**

You can use the predefined template to create a McAfee custom attack to detect an email attachment by the file name.
**Task**

1. In the resource tree, select Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks | Custom Attack Editor.

   The Custom Attack Editor opens with the existing custom attacks listed in the All Custom Attacks tab.

2. Select Attack | New | McAfee Attack Definition | Exploit Attack.

3. On the Templates page, select Detect an e-mail attachment file name and click Next.

4. Enter a Name and Description for the attack.

   *UDS* (User-Defined Signature) is appended at the front automatically when you save the attack. For example, if you name the new attack HTTP Attack XYZ, it appears as *UDS-HTTP Attack XYZ* in the Custom Attack Editor as well as in the attack database when you save the attack.

   ![](image)
   The Manager provides the Attack ID after you save the attack.

5. Select the appropriate Severity for the attack.

6. Select the most appropriate Protection Category for the attack.
   - Skip the Blocking Logic field since it is applicable only for UDP.
   - The protocol for the attack that you are creating is SMTP.
   - You set Target Device Type only in the signatures.

7. From the Direction drop-down list, select the traffic direction that must be defined in this custom attack.
   - Client to Server
   - Server to Client
   - Follow Packet
8 Select the required parameter for Attachment File Name and enter the corresponding value in the text box.

![Add Exploit Attack](image)

**Figure 5-7** Custom attack to detect an email attachment by its file name

9 Click Finish.
10 Right-click the attack that you created in the All Custom Attacks tab, then select Edit. In the Matching Criteria section, the protocol is automatically selected as SMTP.

The signature for the URL that you specified is created automatically with the default values. To edit these values, select the signature and click Edit. You can modify the default values. You can also add more conditions to the signature. From the Edit Exploit Attack window, you can also add more signatures to the attack.

![Details of the McAfee custom attack](image)

**Figure 5-8** Details of the McAfee custom attack

11 To save the McAfee custom attack to the database, in the Custom Attack Editor, select File | Save. Until you save the attack in the database, the NSP Attack ID column in the Custom Attack Editor shows as pending. The Sensor detects this attack after a configuration update.

**Create a McAfee custom attack to detect a DNS query or response**

You can use the predefined template to create a McAfee custom attack to detect a DNS query or response related to a domain name.

**Task**

1. In the resource tree, select Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks | Custom Attack Editor.

   The Custom Attack Editor opens with the existing custom attacks listed in the All Custom Attacks tab.

2. Select Attack | New | McAfee Attack Definition | Exploit Attack.

3. On the Templates page, select Detect a DNS query or response and click Next.
4 Enter a Name and Description for the attack. 

UDS (User-Defined Signature) is appended at the front automatically when you save the attack. For example, if you name the new attack HTTP Attack XYZ, it appears as UDS-HTTP Attack XYZ in the Custom Attack Editor as well as in the attack database when you save the attack.

The Manager provides the Attack ID after you save the attack.

5 Select the appropriate Severity for the attack.

6 Select the most appropriate Protection Category for the attack.

7 In the Blocking Logic field select whether the Sensor should act only on the packet that matched this attack definition or the entire flow.

You set Target Device Type only in the signatures.

8 From the Direction drop-down list, select the traffic direction that must be defined in this custom attack.

- Client to Server
- Server to Client
- Follow Packet

9 Enter the full or the partial Domain Name that the Sensor should detect in a DNS query or response, and then click Finish.

![Custom attack to detect an DNS query or response](image)

**Figure 5-9 Custom attack to detect an DNS query or response**
Right-click the attack that you created in the All Custom Attacks tab, then select Edit.

In the Matching Criteria section, the protocol is automatically selected as DNS.

The signature for the URL that you specified is created automatically with the default values. To edit these values, select the signature and click Edit. You can modify the default values. You can also add more conditions to the signature. From the Edit Exploit Attack window, you can also add more signatures to the attack.

![Figure 5-10 Details of the McAfee custom attack](image)

To save the McAfee custom attack to the database, in the Custom Attack Editor, select File | Save.

Until you save the attack in the database, the NSP Attack ID column in the Custom Attack Editor shows as pending. The Sensor detects this attack after a configuration update.

Create a McAfee custom attack to detect a string in an application running over a custom port

You can use the predefined template to create a McAfee custom attack to detect a string in an application that is running over a custom port.

**Task**

1. In the resource tree, select Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks | Custom Attack Editor.
   
   The Custom Attack Editor opens with the existing custom attacks listed in the All Custom Attacks tab.

2. Select Attack | New | McAfee Attack Definition | Exploit Attack.

3. On the Templates page, select Detect a string in an application running over a custom port and click Next.
4 Enter a **Name** and **Description** for the attack.

*UDS* (User-Defined Signature) is appended at the front automatically when you save the attack. For example, if you name the new attack HTTP Attack XYZ, it appears as **UDS-HTTP Attack XYZ** in the **Custom Attack Editor** as well as in the attack database when you save the attack.

> The Manager provides the **Attack ID** after you save the attack.

5 Select the appropriate **Severity** for the attack.

6 Select the most appropriate **Protection Category** for the attack.

7 If the application is using UDP, then in the **Blocking Logic** field select whether the Sensor should act only on the packet that contained the string or the entire flow.

> You set **Target Device Type** only in the signatures.

8 From the **Direction** drop-down list, select the traffic direction that must be defined in this custom attack.

- **Client to Server**
- **Server to Client**
- **Follow Packet**

9 Enter the **Application Name** and **Application Description**.

Enter the actual application name or the application protocol name in the **Application Name** field. You cannot enter any standard application protocols.

10 Select whether the application is using TCP or UDP and the corresponding port number.

You cannot enter a standard port number.
11 Enter the string that you want the Sensor to detect, then click Finish.

The String to Match field is case-sensitive. That is, the Sensor matches the case when it detects the string.

Figure 5-11 Custom attack to detect a string in an application on a custom port
Right-click the attack that you created in the All Custom Attacks tab, then select Edit. The signature for the URL that you specified is created automatically with the default values. To edit these values, select the signature and click Edit. You can modify the default values. You can also add more conditions to the signature. From the Edit Exploit Attack window, you can also add more signatures to the attack.

![Image of the McAfee custom attack editor](image)

**Figure 5-12 Details of the McAfee custom attack**

To save the McAfee custom attack to the database, in the Custom Attack Editor, select File | Save. Until you save the attack in the database, the NSP Attack ID column in the Custom Attack Editor shows as pending. The Sensor detects this attack after a configuration update.

### Create a McAfee custom attack to detect TCP connection attempts

You can use the predefined template to create a McAfee custom attack to detect TCP connection attempts from specific IP addresses.

**Task**

1. In the resource tree, select Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks | Custom Attack Editor.

   The Custom Attack Editor opens with the existing custom attacks listed in the All Custom Attacks tab.
2 Select Attack | New | McAfee Attack Definition | Exploit Attack.

3 On the Templates page, select Detect a TCP connection attempt from a specific IP address and click Next.

4 Enter a Name and Description for the attack.

   *UDS* (User-Defined Signature) is appended at the front automatically when you save the attack. For example, if you name the new attack HTTP Attack XYZ, it appears as *UDS-HTTP Attack XYZ* in the Custom Attack Editor as well as in the attack database when you save the attack.

   *The Manager provides the Attack ID after you save the attack.*

5 Select the appropriate Severity for the attack.

6 Select the most appropriate Protection Category for the attack.
   *You can skip the Blocking Logic field since it is applicable only for UDP.*
   *You set Target Device Type only in the signatures.*

7 From the Direction drop-down list, select the traffic direction that must be defined in this custom attack.
   *Client to Server*
   *Server to Client*
   *Follow Packet*
8 Enter the IPv4 or IPv6 Source IP Address from which TCP connection attempts are to be detected and click Finish.

The attack is listed in the All Custom Attacks tab.

![Add Exploit Attack](image)

**Figure 5-13** Create a McAfee custom attack to detect TCP connection attempts
Right-click the attack that you created in the All Custom Attacks tab, then select Edit.
The signature for the URL that you specified is created automatically with the default values. To edit these values, select the signature and click Edit. You can modify the default values. You can also add more conditions to the signature. From the Edit Exploit Attack window, you can also add more signatures to the attack.

Figure 5-14 Details of the McAfee custom attack

To save the McAfee custom attack to the database, in the Custom Attack Editor select File | Save.

Until you save the attack in the database, the NSP Attack ID column in the Custom Attack Editor shows as pending. The Sensor detects this attack after a configuration update.

Configure custom reconnaissance attack definition

In the Custom Attack Editor (formerly UDS Editor), you create custom recon attacks using both user-defined exploit attacks and McAfee-defined exploit attacks (in sigset) as component attack. You can also define correlated attacks using these individual attack definitions. For example, UDS attacks that check for URI can be further correlated to test for multiple occurrences in a defined time interval to raise a correlated alert.

To configure Custom Reconnaissance Attack Definition:

- Select Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks | Custom Attack Editor.

The Custom Attack Editor opens with the existing Custom Attacks listed in the All Custom Attacks tab.
• Select Attack | New | McAfee Attack Definition | Reconnaissance Attack.

The Add Reconnaissance Attack window opens.

**Figure 5-15   Add Reconnaissance Attack window**

- Enter a name for the new custom reconnaissance attack.
- Select the Severity, Benign Trigger Probability, and Protection Category from the drop-down lists.
- Set the correlation method for the attack.
  The following correlation methods are available:
  - Brute Force
  - Host Sweep
  - Port Scan
  - Service Sweep
  - Finger Printing
  
  You can select the above options using a single component attack for correlation. Select option e for if you wish to use multiple component attacks for correlation.
- In the Attack field, use the drop-down list to select an attack as a component attack.
  You can select either a McAfee-defined exploit attack or a user-define custom attack as a component attack.
- Set the Threshold Count.
• Set the **Threshold Interval** in seconds.
• Select the **Send Alerts** check box.
• Select the **Add** button to set the **Matching Criteria**.
• Select either **Protocol Only** or **Software Package**, and click **Next**.

![Add Protocols Packages page](image)

**Figure 5-16 Add Protocols Packages page**

• Select the options based on your selection in step 13, and click **Finish**.

![Add Protocol & Package window](image)

**Figure 5-17 Add Protocol & Package window**

You can add more than one matching criteria to your custom reconnaissance attack.

• Review all the settings, and click **Save**.
When traffic passing through the Sensor exceeds the threshold count set for the custom reconnaissance attack within a configured Interval, the Sensor raises an alert to the Manager. You can view the alert in the Threat Analyzer Alerts page. The Alerts Summary View shows the type of attack as either a McAfee Exploit or McAfee Reconnaissance.

The custom reconnaissance attacks are excluded by the IPS engine when:
- One or more of its component attacks is deleted.
- One or more of its component attacks is marked as “Excluded”.

Tasks
- Configure custom attack definition for multiple attacks correlation on page 64

Configure custom attack definition for multiple attacks correlation
You can create custom attack definition for multiple attacks correlation. Do do so, perform the following steps:

Task
1. If you wish to create a custom reconnaissance attack using multiple attacks correlation, select the Correlation Method as Finger Printing.

2. Use the Add button to add multiple attacks as the component attacks for the custom attack that you are creating.
   You must select a minimum of 2 component attacks for multiple attacks correlation.
   You can include a component attack in 10 custom reconnaissance attacks.

3. For the Finger Printing Correlation Method, you can specify the Threshold Interval alone.

Import previously exported custom attacks
If you had exported any McAfee Custom Attacks, then you can import it back into the Manager. Note that this should be the same ZIP file that was created at the time of export. If this ZIP file contains any Snort Custom Attacks, then these are imported as well. This feature enables you to import custom attack definitions created on a different Manager as well as the McAfee-defined custom attacks.

When you import the ZIP file containing the exported attack definitions, only the attacks not present in the Manager are imported.

To import the previously exported McAfee Custom Attacks:
Task

1. In the Custom Attack Editor, go to File | Import | Custom McAfee Attack.

2. Browse to the location of your saved McAfee Custom Attack ZIP file.

3. Click Open.

4. Select Save from the File menu of the Custom Attack Editor to save the imported attacks in the Manager Server. Only when you save the attacks in the Manager server, they are included in the policies.

5. Verify if the attack is included in the policies.

6. Push the attack with its signatures to the Sensors for attack detection to happen.

See also

Viewing a policy to verify inclusion of the attack on page 132

Importing McAfee-defined custom attacks

To quickly address the latest threats, McAfee periodically provides attack definitions before they are incorporated in official signature sets. These McAfee-defined custom attacks (ID-UDS) are automatically deprecated when they are incorporated in a regular signature set.

When an ID-UDS is deprecated, the Manager generates an Informational alert that you can view in the Threat Analyzer.

You can not edit the attributes of an ID-UDS using the Custom Attack Editor.

You can export and import an ID-UDS just like you would with any other Custom Attacks.

Work with packet search protocol

This section deals with creating a packet search protocol and creating a signature using your new packet search protocol.
Create a packet search protocol

Not all traffic can be supported by previously defined protocol specifications. Thus in some cases you may need to define a different means of separating out certain protocol traffic. The Packet Grep Protocol feature (Packet Search > Manage Packet Search Protocol) enables you to create pattern matching searches in a particular protocol traffic that is not separately parsed.

Packet searching is commonly known as packet grepping. The term grep refers to the Unix command "g/re/p," which executes a global search for a regular expression, and prints the lines that contain pattern matches. In essence, a protocol packet grep is a search for a fixed pattern in either the request or response of an unsupported protocol flow.

Protocols, such as HTTP and FTP, have defined specifications and destination ports. For example, HTTP primarily uses port 80, but is also common on port 8080. A Sensor monitoring traffic identifies the HTTP traffic by the protocol's common traits, and checks the traffic against standardized (RFC) HTTP traffic specifications. If packets are determined to be suspicious through signature tests, an alert is sent from the Sensor to the Manager. In the event that a Sensor does not recognize the protocol of a monitored transmission through signature, anomaly, or packet search tests, the packets are sent to a general state machine and examined against user-configured pattern-matching checks. If a pattern is matched, an alert is raised.

For example, port 1387 on one of your Payroll servers is receiving undetermined protocol connection requests atop TCP. You have confidential information on this server, so protecting your data from suspicious searches or requests is crucial. You can create a packet search instance in requests via TCP to port 1387, then create a McAfee Custom Attack that defines a regular expression pattern that alerts upon being matched.
Several packet search protocol instances are provided with Network Security Platform. These applications use protocols that are not defined by an RFC nor by a defined McAfee protocol specification. Some of these instances, such as pcAnywhere and NetMeeting, represent legitimate applications that are allowed by network policy within/into some networks, but which may be used for malicious purposes.

You cannot delete any of the McAfee-defined packet grep protocols.

![Packet Grep Protocols window](image)

**Figure 5-19  Packet Grep Protocols window**

**See also**

*Create a signature on page 44*

**Create a packet search protocol instance**

Several packet search protocol instances are provided with Network Security Platform. You can create a packet search protocol instance. These applications use protocols that are not defined by an RFC nor by a defined McAfee protocol specification.

**Task**

1. In the Custom Attack Editor, select **Packet Search** | **Manage Packet Search Protocol**.
   
   The Packet Grep Protocols dialog box opens.

2. Click **Add**.

3. Type a name. This name is listed in the "Select Protocol" step during signature creation.

4. Type a verbose name. This name is for reference purposes.

5. Select the transport protocol as either **TCP** or **UDP**.
6 Select one or both of the following for packet grep (that is, search) operations:
   • Process Request: enables a search in request traffic.
   • Process Response: enables a search in response traffic.

7 Type a port number in the Port Number field, then click Add. Repeat for multiple port numbers.

   You cannot add a port number that is already used by a defined protocol, such as 21 (FTP) or 80 (HTTP). Also, if you create a packet grep instance for port 888, you cannot create another packet grep instance for that port.

8 Click OK when finished.

9 Create a new signature instance that utilizes your created packet grep instance.

### Create a signature using your new packet search protocol

**Before you begin**

To create a McAfee Custom Attack to find attacks using your custom packet search protocol, do the following:

Follow the steps in under Creating a McAfee Custom Attack to create the attack definition. Remember to select tcpip-machine as the impact package.
**Task**

1. Add a new signature to the attack.

2. Add a comparison with at least one condition. Do the following:
   a. Click ADD under Conditions. Condition 1 appears in the field below.
   b. Click Condition 1 so that it is highlighted.

![Figure 5-20 Creating a condition](image)

3. Click AND under Comparisons. A dialog box opens.

![Figure 5-21 Adding comparators](image)

4. Select Packet Grep Protocol from the Comparison List and then click Next.
5. Select the protocol name that you defined from the Protocol List, then click Next.
6. Configure the fields for the comparison you have chosen.
   1. Select the signature search direction: for request, select pktsearch-<protocol>-req-text; for response, select pktsearch-<protocol>-rsp-text.
   2. From the drop-down list in the Regular Expression to Match section, select the matching criteria as Equals or Does NOT Equal based on the matching criteria required for the string pattern entered.
   3. Add a pattern to match using Regular Expression to Match.
   4. Optionally, click the Case Insensitive check box if you want the pattern to be matched regardless of [letter] case.
5 Click **Validate** to verify that your pattern is valid.

6 Click **Finish** when done with the Configure Comparison fields. Your comparison appears under **Condition 1**.

![Configure Comparison window](image)

**Figure 5-22 Configure Comparison window**

3 Click **Save** in the Add McAfee Attack interface.

4 Select **Save** from the File menu of the Custom Attack Editor to save the attack and the signature in the Manager server database.

---

### Regular expression language

This chapter describes the proper method of writing pattern-matching signatures using the Regular Expression Language supported for McAfee Custom Attacks.

Signature creation requires a working knowledge of protocols and the data they carry across your network, as well as an ability to write a proper signature using a regular expression language. This information is intended for advanced network security personnel who have experience with McAfee Custom Attack creation.

Signature creation is based on a set of expressions in a defined syntax, also known as regular expressions.

**See also**

*Create a signature on page 44*
Regular expression syntax
All expressions are generated from the following set of rules for pattern representation, and only from these rules:

Printable characters
You can use any alphanumeric characters for basic string matching purposes.

• lower case 'a' through 'z'
• upper case 'A' through 'Z'
• numerals '0' through '9'

Thus, typing "confidential" and marking the test as Case-Insensitive denotes that either "Confidential" or "confidential" will match. Likewise, the string "007forever" marked as Case-Sensitive denotes a search for the exact phrase "007forever".

Special characters
Special characters are those that are non-alphanumeric/punctuation. They require a special representation so that they are visible and not translated as formatting. A "\r" is a carriage return, part of what makes your cursor go to the next line.

<table>
<thead>
<tr>
<th>Action or Symbol</th>
<th>ASCII</th>
<th>Type as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>carriage return</td>
<td>CR</td>
<td>\r</td>
</tr>
<tr>
<td>newline</td>
<td>NL or LF</td>
<td>\n</td>
</tr>
<tr>
<td>horizontal tab</td>
<td>HT</td>
<td>\t</td>
</tr>
<tr>
<td>vertical tab</td>
<td>VT</td>
<td>\v</td>
</tr>
<tr>
<td>formfeed</td>
<td>FF</td>
<td>\f</td>
</tr>
<tr>
<td>backspace</td>
<td>BS</td>
<td>\b</td>
</tr>
<tr>
<td>bell char</td>
<td>BS ??</td>
<td>\a</td>
</tr>
<tr>
<td>match any single character</td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Read as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>first line\nsecond line</td>
<td>first line second line</td>
</tr>
</tbody>
</table>

Hexadecimal values
Hexadecimal values can be represented as two alphanumeric characters (a to f, and A to F, and 0 to 9) preceded by "\x"; thus "\x00" is a proper hex string. The range is from \x00 to \xFF, and matching must be case-insensitive.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\x90\x90</td>
<td>Example of hex used to represent binary; often used in shellcode attacks.</td>
</tr>
</tbody>
</table>

Character class
To search for a range of characters, define the range using a hyphen between characters [A - Z].
<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1-9]</td>
<td>Any digit from 1 to 9 (1, 2, 3, 4, 5, 6, 7, 8, 9)</td>
</tr>
<tr>
<td>[a-j]</td>
<td>Any lowercase letter from a to j.</td>
</tr>
<tr>
<td>[A-H]</td>
<td>Any capitalized letter from A to H.</td>
</tr>
</tbody>
</table>

A search such as [a-F] is not valid.

### Optional

To make a character or pattern optional, use parentheses with a trailing question mark: `(string)?`.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code red (virus)?</td>
<td>The search for the word &quot;virus&quot; is optional. This pattern matches if either &quot;code red&quot; or &quot;code red virus&quot; is detected.</td>
</tr>
</tbody>
</table>

### Alternation

To search using alternation, use the pipe (|) character to separate strings (a|b).

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(hi</td>
<td>hello)</td>
</tr>
<tr>
<td>(su</td>
<td>soo)per</td>
</tr>
</tbody>
</table>

### Repeating

To repeat, append a number in brackets to the pattern: for example, `(string){n}` means the string must be repeated `n` times for matching purposes.

- You must enclose the string in parentheses, and the repeating value must be enclosed in brackets.
- The number in braces must be greater than 0 (n>0).

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1){5}=11111</td>
<td>A string of five consecutive 1s without spaces must match.</td>
</tr>
<tr>
<td>(text){3}=texttexttext</td>
<td>A string consisting of the word text repeated three times without spaces must match.</td>
</tr>
</tbody>
</table>

To set a minimum and maximum, represent the equation as `(string){3,5}`

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1){3,5}=111</td>
<td>A string of three, four, <strong>OR</strong> five consecutive 1s without spaces must be match.</td>
</tr>
</tbody>
</table>

### Anchors

To specify a string that starts or ends a line, use the `^` (carat) or `$` (dollar) characters. Using both `^` and `$`, you can also specify a string that makes up an entire line—start to end.

- Start of line anchor: `^(string)`
- End of line anchor: `(string)$`
### Pattern Description

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>^Hi</code></td>
<td>The first word of a line must be &quot;Hi&quot;.</td>
</tr>
<tr>
<td><code>bye$</code></td>
<td>The last word of a line must be &quot;bye&quot;.</td>
</tr>
<tr>
<td><code>^Hacked by J03$</code></td>
<td>&quot;Hacked by J03&quot; forms an entire line.</td>
</tr>
</tbody>
</table>

### Reserved characters

The following characters are reserved for regular expression control. They must be escaped; that is, prefixed with backslash (\) when they appear as part of a normal matching pattern (that is, not part of an expression), except where noted otherwise.

#### Name | Symbol | Proper Syntax
--- | --- | ---
backslash | \ | \ \ 
dot (period) | . | . 
pipe | | \ 
dash (hyphen) | - | - 
carat | ^ | ^ 
dollar symbol | $ | $ 
open paren | ( | ( 
close paren | ) | ) 
open bracket | [ | [ 
close bracket | ] | ] 

#### Pattern | Read as:
--- | ---
(text) | text 
\((text)\) | \(text\) 
www.McAfee.com | www*McAfee*com, where * is a single wildcard 
www\,McAfee\,com | www.McAfee.com 

### Limitations of a custom reconnaissance attack

The limitations of Custom Reconnaissance attack are listed below:

- The maximum threshold count for Custom Reconnaissance attack is 255 for single-AID (type: port scan, host sweep, service sweep, and brute force).
- A component attack can only be included in 10 Custom Reconnaissance attacks.
- A minimum of 2 component attacks need to be added for a multiple-AID (type: fingerprinting) Custom Reconnaissance attack.
- Maximum 5 component attacks can be added in a multiple-AID (type: fingerprinting) Custom Reconnaissance attack.
- Maximum 300 Reconnaissance attacks (McAfee and Custom) can be pushed to the Sensor.
- Maximum 300 component attacks can be pushed to the Sensor.
- Maximum 150 Custom Reconnaissance attacks can be added to the Custom Attack Editor.
Managing custom attacks
Limitations of a custom reconnaissance attack
This section explains the structure of a Snort Custom Attack. It also provides some information on how to construct valid Snort rules.

**Contents**
- Structure of a Snort custom attack
- Structure of a snort rule

## Structure of a Snort custom attack

A Snort Custom Attack is made up of one Snort rule. To define a Snort Custom Attack, you either write the Snort rule directly in the Custom Attack Editor or import it into the Editor from a file. The Sensor automatically creates the Snort Custom Attack record for each rule that you write or import.

When the Sensor creates the attack for a Snort rule, it automatically defines the values for fields such as attack name and attack severity. The following section explains how the Sensor defines the attack-level values for a Snort Custom Attack:

For the Attack Name, the format that the Sensor uses for the name is **SNORT:<the text specified for the msg rule option in the rule>(<SID>).** So this name is modified accordingly if you modify the msg text or the SID of the rule.

- **msg rule option and a unique SID are mandatory for a Snort Custom Attack.**

Blocking is set to attack packet for all Snort Custom Attacks. You cannot modify this.

In the Impact tab, the Severity is based on the classtype keyword or the priority tag. Each classtype has a default priority defined. Typically, the classtypes with the default priority values are defined in the classification.config file. If a rule has both classtype keyword and priority tag, then the severity is based on the priority tag.

The Sensor assigns the severity for a Snort custom attack based on the following:

- A priority 1 Snort attack definition is assigned a severity of High.
- A priority 2 Snort attack definition is assigned a severity of Medium.
- A priority of 3 or higher is assigned a severity of Low.

When you modify the classtype or the priority tag value of a rule, the Sensor will also modify the severity accordingly. This means that when you save the attack to the Sensor server subsequently, the rule sets (and policies) in which the attack was included may change.

For the Exploit category, tcpip-machine is selected as the package with any as the OS. You cannot add, delete, or modify any impact package or protocol for Snort Custom Attacks.
Structure of a snort rule

This section discusses the basic structure of a Snort rule and the Snort rule elements that are supported in Network Security Platform. This section contains information on the syntax that you should use in Network Security Platform for each element. You can also find an example rule for each element, wherever applicable.

How you combine various Snort rule elements to form a rule depends on your security requirements. This information is out of scope of this document because the requirements can vary from network to network.

The Snort rules that you plan to use in Network Security Platform should conform to Snort rules language.

A Snort rule in Network Security Platform can run into multiple lines. A typical Snort rule has two logical sections - rule header and rule options.

The following is an example of a simple but valid Snort rule.

```snort
alert tcp any any -> 10.1.1.1 80 (msg:"a sample rule"; content:"hello world"; sid:10000; priority:3;)
```

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rule header section</td>
</tr>
<tr>
<td>2</td>
<td>Rule options section</td>
</tr>
</tbody>
</table>

As shown above, the rule header section starts at the beginning of the rule and ends at the opening parenthesis. This section contains the following:

- Action - in the sample above it is alert.
- Protocol - in the sample, it is tcp.
- Source and destination IP addresses - in this case the source is any and the destination is 10.1.1.1
- Source and destination ports - in this case the source port is "any" and the destination port is 80.

The rule options section is enclosed within the parentheses. This section contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken. In the rules options section, the words ending in a colon are option keywords.

In Network Security Platform, the rule options section should have at least the msg and the sid keywords with values. Rule options enable you accurately define the attack traffic that the Sensor should look for.

All the elements that make up a rule must be true for the Sensor to raise an alert and take the defined response action. You specify the response action for the Snort Custom Attack in the IPS Policy Editor.

**Rule header section**

In the rule header section, you define:

- action the Sensor needs to take when it sees the attack traffic.
- transport layer protocol of the attack traffic
• source and destination IP for the attack traffic
• communication port at both the source and destination of the attack
• direction of the traffic that the Sensor should inspect.

The items listed above are discussed in detail in the subsequent sections.

**Rule action**

The action specifies how the Sensor should respond when it detects traffic as defined in the rule. In Network Security Platform, the only action that you can specify in a Snort rule is to send an alert to the Manager. The procedure to configure other response actions such as packet log and packet drop is the same as that of the regular McAfee attacks. That is, use the IPS Policy Editor to configure these actions.

**Protocols**

In the protocols section you specify the protocol to which the Sensor should apply the rule. The options are TCP, UDP, ICMP, and IP.

**IP addresses**

You need to specify the IP address of the source and target for the attack traffic. The available options are listed below:

- You can use the keyword "any" to specify a wildcard IP address.
- You can use CIDR blocks to specify a range of IP addresses. Example: 172.16.230.0/24
- You can use the negation operator. Example: !172.16.230.0/24. If you specify this example as the source, then you are indicating that the source of the attack traffic must not be from the 172.16.230.0 network.
- You can specify a list of IP addresses by specifying them inside square brackets with comma as the separator. Example: [172.16.230.0/24, 172.16.231.0/24]
- You can use variables to specify particular IPs or a range of IPs.

**Port numbers**

You need to specify the port numbers used at the source and the target for the attack traffic. The available options are listed below:

- You can use the keyword "any" to specify a wildcard port number.
- You can specify a port range using the range operator, colon. Example: 1:1024. This indicates that the port must be between 1 and 1024, both inclusive.
- Example: 100: This indicates the port must be equal to 100 or more.
- Example: :500. This indicates that the port must be less than or equal to 500.
- You can use the negation operator. Example: !100:200. This indicates that the port number should not be between 100 and 200.
- You can use variables to indicate ports.
**Direction operator**

The direction operator `->` indicates the direction of the traffic that the Sensor should apply the rule to. The IP address and port numbers on the left side of the direction operator are that of the source and the ones on the right side are that of the destination of the attack traffic. You can also use the bidirectional operator, which is indicated by `<->`. This means that either side of the operator can be the source or the destination. This can be handy when analyzing traffic in both sides, such as in telnet or POP3 sessions.

**Rule options section**

The rule options section is the more critical section of a Snort rule. The rule options are to be separated by a semicolon. In a rule option, the keyword and argument are separated by a colon. Rule options fall into one of the following categories:

- General: These options contain the metadata for the rule but have no effect on attack detection.
- Payload: These options look for data inside the packet payload and can be inter-related.
- Non-payload: These options look for non-payload data.
- Post-detection: These options indicate what happens when an attack is detected. These options are not relevant in Network Security Platform. Instead you can configure the response actions in the IPS Policy Editor.

**General rule options**

**msg**

In the msg rule option, you indicate the alert message. It is a simple text string that utilizes the `\` as an escape character to indicate a discrete character such as a semi-colon.

Syntax:

```
msg: "<message text>";
```

When you create a Snort Custom Attack in the Custom Attack Editor, the Manager assigns the attack and rule name, which you cannot edit directly. The format of the name is as below:

Snort: <text entered for msg option> (sid: <sid value>)

The argument of the msg option features in the attack and rule name that the Manager assigns. Changing the msg argument or sid value and saving the rule will change the attack and rule name accordingly.

**reference**

In a rule, you can use the reference keyword to include references related to the corresponding attack traffic. For example, it could be the bugtraq or CVE id of known vulnerabilities. It could also be the address of a website that provides the details of the attack.

The following are the supported systems that you can specify in the reference option:

- bugtraq
- CVE
- McAfee
- Arachnids
- URL
- URL

Syntax: `reference:<id system>, id;`
Examples:

reference: bugtraq, 514;

You can view the references that are currently available in the Manager.

**gid**

gid stands for generator id and is not relevant in Network Security Platform. So, do not use them in your rules. If a rule that you imported contains a gid, then this information is displayed in the Conversion Notes section of the rule. See the highlighted section of the graphic below.

![Figure 6-1 Edit Snort Attack window](image)

**sid**

sid stands for Snort rule ID. You must specify a unique sid for each Snort rule in the Manager. For Snort Custom Attacks that failed to import, the Manager assigns -1 as the sid.

Syntax:

sid:<sid value>;

**rev**

This enables you to identify any revisions of an existing rule. It is not a mandatory option, but Snort Custom Attacks without the rev option are converted with a warning. When you edit a Snort Custom Attack in the Manager, you can modify the rev value for your reference.
If you want to re-import a Snort Custom Attack with the same sid, then you can change the rev value and import it. The Manager overwrites the existing Snort Custom Attack with the one that you last imported.

Syntax:
rev:<revision integer>;

classtype
This is one of the options that you can use to categorize a Snort rule. In the Snort rules provided by Snort.org, the classtypes with the default priority are defined in the classification.config file. You can use these classtypes by importing the classification.config file into the Manager. Alternatively, you can define your own in a .config file and import it into the Manager. If you are defining your own classtype, then make sure you also assign a priority to the classtype.

A rule must have a classtype or a priority.
You can view the classification types that are currently available in the Manager.

Syntax for using a classtype in a rule is as follows. This assumes that the class name is available in the Manager database:
classtype: <class name>;

Syntax (for defining a classtype in a .conf file):
config classification: <class name>,<class description>,<default priority>
Example:
config classification: attempted-user,Attempted User Privilege Gain,1

priority
Using the priority option you can assign a severity to the rule. This overrides the default severity defined in the classtype.

The Manager assigns the severity for a Snort custom attack based on the following:
• A priority 1 Snort attack definition is assigned a severity of High.
• A priority 2 Snort attack definition is assigned a severity of Medium.
• A priority of 3 or higher is assigned a severity of Low.

Syntax: priority: <integer value>;

metadata
This option enables you provide information about the rule. This is a free-form option, which means you can use anything as long as it conforms to the Snort Rules Language syntax.

Syntax: metadata: key1 value1, key2 value2;

Payload rule options
This section describes the options and conditions that you can apply to the payload in TCP, UDP, or ICMP traffic. Note that some of the options work differently in Network Security Platform when compared to Snort. Also, some options are not supported.
The payload rule options discussed in this section include content modifier and preprocessor rule options. You use content modifiers to change the way content option works. These are specifications that the Sensor should apply on the content data before it looks for a match.

**Content**

This enables you to specify the content that a Sensor should look for in the packet payload. If the Sensor finds traffic matching what you specify in the content, then it raises an alert and also takes the defined response action such as packet log or packet drop.

The option data for the content keyword can contain text, binary (hex values), or mixed text and binary data. The binary data must be enclosed within the pipe (|) character and represented as bytecode.

To reduce the chance for false positives, you can define multiple content in one rule. You can use the negation operator (!). If you precede the content with !, then Sensor raises an alert if traffic does not contain the content that you specified.

- The following characters must be escaped inside a content rule:
  - : (colon)
  - ; (semi colon)
  - \ (backward slash)
  - " (double quotes)
- Though you can specify even a single-byte token, the longer the content value, the accurate the detection will be. The maximum length of a content or uricontent for I-series Sensors is 96 bytes and 256 bytes in case of M-series and NS-series.
- Make sure the content option does not contain any generic values identified by McAfee (some examples are listed below). Such rules can severely impact Sensor performance.
  - GET
  - POST
  - Host:
  - User-Agent

**Syntax:**

```plaintext
content:"<content string>";
```

**Example rules:**

- ```plaintext
  alert tcp any any -> 10.1.1.1 80 (msg:"command dot exe attempt"; content:"cmd.exe"; flow:to_server; sid:2001; priority: 1;)
  ```
- ```plaintext
  alert tcp any any -> 10.1.1.1 80 (msg:"command dot exe attempt"; content:"|63 6D 64 2E 65 78 65|"; flow:to_server; sid:2001; priority: 1;)
  ```
- ```plaintext
  alert tcp any any -> 10.1.1.1 80 (msg:"command dot exe attempt"; content:"|63| m |64 2E 65 78 65|"; flow:to_server; sid:2001; priority: 1;)
  ```

This section explains the content modifiers, which you can use to change the way content option works. The content modifiers are specifications that the Sensor should apply on the content data before it looks for a match.
**nocase**

Use this if you want the Sensor to look for the content data regardless of the case.

Syntax: `nocase;`

Example rule: `alert tcp any any -> 10.1.1.1 80 (msg: "command dot exe attempt"; content: "cMd.eXe"; nocase; flow:to_server; sid:2002; priority: 1;)`

**rawbytes**

Use this if you want the Sensor to consider the content data as raw data. Then Sensor matches the content data against the non-normalized traffic.

Syntax: `rawbytes;`

For example, `http://www.example.com/big%20cars/blue%20colored%20cars/pictures.htm`

will normalize to `http://www.example.com/big cars/blue colored cars/pictures.htm`.

If you want the Sensor to find "blue%20color" in the traffic before normalization (that is, as seen in the "wire"), then your rule could be:

`alert tcp any any -> 10.1.1.1 80 (msg: "example rule for rawbytes"; content:"blue%20color"; rawbytes; flow:to_server; sid:2003;priority:1;)`

If you do not specify rawbytes for the same rule, the Sensor will look for "blue%20color" in the normalized traffic, which it would never find.

**depth**

Use this to specify how deep in the payload should the Sensor look for the content data. For example, a depth of 5 means that the Sensor should look for the specified pattern within the first 5 bytes of the payload.

Content option should precede depth in the rule.

Syntax: `depth: <byte count>;`

Example rule: `alert tcp any any -> any any (msg: "example rule for depth"; content:"blue color"; depth:30; sid:2003;priority:1;)`

**offset**

Use this to specify where a Sensor should begin its search within the payload. An offset of 5 means that the Sensor start its search after the first 5 bytes of the payload.

Content keyword should precede offset in the rule.

Syntax: `offset: <byte count>;`

Example rule: `alert tcp any any -> any any (msg: "example rule for offset"; content:"blue color"; offset:5; sid:2004;priority:1;)`

**distance**

Use this to specify how far into a packet, the Sensor should ignore before it starts the search for the specified pattern relative to the end of the previous pattern match.

This is same as that of offset except that it is relative to the end of the last successful pattern match instead of the beginning of the packet.

Content key word should precede distance.

Syntax: `distance: <byte count>;`
Example rule: alert tcp any any -> any any (msg:"example rule for distance"; content:"red color"; content:"blue color"; distance:2;sid:2005;priority:1;)

For this rule, the Sensor looks for "red color" in the payload. If found, skips the 2 immediate bytes, and then looks for "blue color". If this string is also found, it raises an alert.

**within**

In the case of this modifier, the Sensor checks if the content is within the specified bytes from the previous content match.

Content key word should precede within. Typically, this modifier is used along with distance.

**Syntax:** within: <byte count>;

Example rule: alert tcp any any -> any any (msg:"example rule for within"; content:"blue"; content: "red"; within:5;sid:2006;priority:1;)

For this rule, the Sensor raises an alert only if it finds "red" within 5 bytes after "blue".

**http_client_body**

Use this modifier if you want the Sensor to look into only the normalized http client body.

Note the following:

- Content should precede http_client_body, and it applies only to the immediately preceding Content.
- If you use http_client_body with offset or depth, then these options are calculated from the beginning of the http payload. Consider the example rule below:

  alert tcp any any -> any any (msg: "example rule for http_client_body"; content:"red"; http_client_body; depth:30; sid:2007;priority:1;)

  The Sensor looks for "red" within 30 bytes from the beginning of the http payload and not from the beginning of the http client body.
- Because this modifier works only for the normalized data, you cannot use it with rawbytes for the same content.

Example rule:

alert tcp any any -> any any (msg: "example rule for http_client_body"; content:"red"; content:"blue"; http_client_body;sid:2007;priority:1;)

For this rule, the Sensor looks for "red" in the entire payload and for "blue" only in the normalized HTTP request.

**http_cookie**

The http_cookie modifier is not supported in Network Security Platform.

**http_header**

In this case, the Sensor checks for the content only in the normalized HTTP request.
Note the following:

- Content should precede `http_header`, and it applies only to the immediately preceding Content.
- If you use `http_header` with `offset or depth`, then these options are calculated from the beginning of the payload. Consider that you created a rule as shown below:
  ```
  alert tcp any any -> 10.1.1.1 80 (msg: "example rule for http_header"; content:"user"; http_header; depth:100; sid:2008;priority:1;)
  ```
  The Sensor looks for "user" within 100 bytes from the beginning of the http payload.

- You cannot apply the `http_header` and `rawbytes` on the same content.

**Syntax:** `http_header;`

**Example rule:**
```
alert tcp any any -> any any (msg:"example for http_header"; content:"red"; content:"user";http_header;sid:2008;priority:1;)
```
In this rule, the Sensor looks for "red" in the entire payload and for "blue" just in the normalized HTTP header part of the payload.

**http_method**

In this case, the Sensor checks for the content only in the normalized HTTP method section within a HTTP request.

Note the following:

- Content should precede `http_method`, and it applies only to the immediately preceding content.
- If you use `http_method` with `offset or depth`, then these options are calculated from the beginning of the http payload and not within a specific http field. Consider that you created a rule as shown below:
  ```
  alert tcp any any -> any any (msg: "example rule for http_method"; content:"post"; http_method; depth:5; sid:2009;priority:1;)
  ```
  The Sensor looks for "post" within 5 bytes from the beginning of the http payload.

- You cannot apply the `http_method` and `rawbytes` to the same content.

**Syntax:** `http_method;`

**Example rule:**
```
alert tcp any any -> any any (msg:"example for http_method"; content:"red"; content:"PUT";http_method;sid:2009;priority:1;)
```
For this rule the Sensor looks for "red" in the entire payload and whether the HTTP method is PUT.

**http_uri**

In this case, the Sensor checks for the content only in the normalized request URI section.

Note the following:

- Content should precede `http_uri`, and it applies only to the immediately preceding Content.
- Functionally, using `http_uri` is same as using the `uricontent` option explained later in this section.
• If you use http_uri with offset or depth, these options are calculated from the beginning of the http payload. Consider that you created a rule as shown below:

```
alert tcp any any -> any any (msg: "example rule for http_uri"); content:"red"; http_uri; depth:50; sid:2009; priority:1;)
```

The Sensor looks for "red" within 50 bytes from the beginning of the http payload.

• You cannot apply http_uri and rawbytes on the same content.

**Syntax:** http_uri;

**Example rule:** alert tcp any any -> any any (msg:"example for http_uri"; content:"red"; content:"blue"; http_uri; sid:2009; priority:1;)

In this rule, the Sensor looks for "red" in the entire payload; "blue" just in the normalized request URI.

**http_raw_cookie**

This searches the extracted unnormalized cookie header field of a HTTP request or a HTTP response. Since this is a content modifier to the previous content, there must be a content in the rule preceding the http_raw_cookie rule option.

```
In a Snort custom attack, you cannot use http_raw_cookie modifier with rawbytes or http_cookie modifiers for the same content.
```

**Syntax:** http_raw_cookie;

**Example rule:** alert tcp any any -> any 80 (msg: "example rule for http_raw_cookie"; content:"red"; content:"blue"; http_raw_cookie;sid:2099; priority:1;)

This rule searches only for blue in the extracted unnormalized cookie header field of a HTTP request.

**http_raw_header**

This searches the extracted unnormalized header fields of a HTTP request or a HTTP response. Since this is a content modifier to the previous content, there must be a content in the rule preceding the http_raw_header rule option.

```
In a Snort custom attack, you cannot use http_raw_header modifier with rawbytes or http_cookie modifiers for the same content.
```

**Syntax:** http_raw_header;

**Example rule:** alert tcp any any -> any 80 (msg: "example rule for http_raw_header"; content:"red"; content:"blue"; http_raw_header;sid:2199;priority:1;)

This rule searches only for blue in the extracted unnormalized header fields of a HTTP request or HTTP response.

**http_raw_uri**

This searches the unnormalized request URI field. Since this is a content modifier to the previous content, there must be a content in the rule preceding the http_raw_uri rule option.

```
In a Snort custom attack, you cannot use http_raw_uri modifier with rawbytes or http_cookie modifiers for the same content.
```

Syntax: http_raw_uri;

Example rule: alert tcp any any -> any 80 (msg: "example rule for http_raw_uri"; content:"red"; content:"blue"; http_raw_uri;sid:2499;priority:1;)

This rule searches only for blue in the unnormalized URI.

http_stat_code

This searches the extracted status code field from a HTTP response. Since this is a content modifier to the previous content, there must be a content in the rule preceding the http_stat_code rule option.

In a Snort custom attack, you cannot use http_stat_code modifier with rawbytes modifier for the same content.

Syntax: http_stat_code;

Example rule: alert tcp any any -> any 80 (msg: "example rule for http_stat_code"; content:"red"; content:"200"; http_stat_code;sid:7199;priority:1;)

This rule searches for status 200 in the extracted status code field of a HTTP response.

http_stat_msg

This searches the extracted status message field from a HTTP response. Since this is a content modifier to the previous content, there must be a content in the rule preceding the http_stat_msg rule option.

In a Snort custom attack, you cannot use http_stat_msg modifier with rawbytes modifier for the same content.

Syntax: http_stat_msg;

Example rule: alert tcp any any -> any 80 (msg: "example rule for http_stat_msg"; content:"red"; content:"OK"; http_stat_msg;sid:6199;priority:1;)

This rule searches for OK in the extracted status message field of a HTTP response.

fast_pattern

This option is not relevant for a Snort rule in Network Security Platform, and is not supported. If a rule that you imported contained this option, it is displayed in the Conversion Notes section of the rule. The Sensor ignores this option when it checks the traffic against this rule.

uricontent

Use this keyword to search in the normalized request URI field. Note that if the uricontent value has anything that is normalized, then the rule will return negative. For example, if the value contains %20, then the Sensor does not raise an alert because it will check for %20 in the normalized uricontent, which it will never find. To check for non-normalized content, consider using the content option with rawbytes.
Note the following:

- This is same as using content with http_uri.
- If you use uricontent with offset or depth, these options are calculated from the beginning of the http payload. Consider that you created a rule as shown below:
  
  ```
  alert tcp any any -> any any (msg: "example rule for uricontent"; uricontent:red; depth:50; sid: 2009;priority:1;)
  
  The Sensor looks for "red" within 50 bytes from the beginning of the http payload.
  ```

- You cannot use rawbytes with uricontent.

  ```
  Syntax: uricontent:[!]<content string>;
  
  Example rule: alert tcp any any -> any any (msg:"example for uricontent"; content:"red"; uricontent:"hello+world"; sid:2033; priority:1;)
  
  In this rule, the Sensor looks for "red" in the entire payload; "hello+world" just in the normalized request URI.
  ```

**urilen**

This is a condition that enables you to specify the exact, minimum, or maximum lengths, or range of URI lengths to match.

**Syntax:**

```
urilen: int<>int;
urilen: [<,>] <int>;
```

**Examples:**

```
urilen:5
Matches URIs that are 5 bytes in length.

Example rule: alert tcp any any -> any any (msg:"example for urilen"; content:"red"; content:"blue"; urilen:5; sid:2009; priority:1;)

For this rule, the Sensor checks for the strings "red" and "blue" in the entire payload and also sees if the request URI is exactly 5 bytes in length.

urilen: < 5
Matches URIs that are less shorter than 5 bytes.

urilen: 5<>10
Matches URIs that are >= 5 bytes and <= 10 bytes.
```

**isdataat**

This condition enables you to check if payload has data at a specific location. You can also look for data relative to the end of the previous content match.

**Syntax:**

```
isdataat:[!]<int>[, relative];
```
Example rule: alert tcp any any -> any any (msg:"example for isdataat"; content:"red"; isdataat:20,relative; content:"blue"; sid:2010; priority:1;)

For this rule, the Sensor looks for "red" in the payload, then checks if there are at least 20 bytes as well as "blue" after "red".

In a Snort custom attack, you cannot use isdataat with rawbytes modifier for the same content.

**pcre**

You can use Perl Compatible Regular Expressions (PCRE) with the related modifiers in the Snort rules. If you use PCRE in a rule, then it should be preceded by a Content. The Content is the triggering token for PCRE options following it. That is, the Sensor checks for the PCRE part of the rule only when the traffic is positive for the preceding Content. Do not use negation operator on the triggering token.

PCRE options are resource-intensive on the Sensor. McAfee recommends that you check if there are other options to achieve the same result.

**Syntax:**

```
pcre:[]"/(<regex>)/[ismxAEGUPHMCOIDKYS]"
```

- `/` is the only supported delimiter.
- For the PCRE options, by default, the Sensor considers only the first 256 bytes from the beginning of the triggering token.
- When you import or validate the Snort rules in the Manager, you can know the unsupported PCRE options due to which the rules failed to convert successfully.

The following table explains the PCRE-related options supported in Network Security Platform.

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>i</code></td>
<td>The Sensor ignores the case of the corresponding string.</td>
</tr>
<tr>
<td><code>s</code></td>
<td>Use this to consider new lines in the dot meta-character.</td>
</tr>
<tr>
<td><code>m</code></td>
<td>This is used with the anchors (^ and $).</td>
</tr>
<tr>
<td></td>
<td>Use <code>m</code> with <code>^</code> if you want the Sensor to check for the string immediately after a new line as well as at the beginning of the buffer.</td>
</tr>
<tr>
<td></td>
<td>Use <code>m</code> with <code>$</code> if you want the Sensor to check for the string immediately before a new line as well as at the end of the buffer.</td>
</tr>
<tr>
<td><code>x</code></td>
<td>Use this if you want the Sensor to ignore any empty space characters in the buffer unless it is escaped or inside a character class.</td>
</tr>
<tr>
<td><code>A</code></td>
<td>The Sensor checks if the string is at the beginning of the buffer. This is the same as <code>^</code>.</td>
</tr>
<tr>
<td><code>E</code></td>
<td>This is used with <code>. Without </code>E<code>, </code>.` also matches immediately before the final character if it is a new line but not before any other new lines.</td>
</tr>
<tr>
<td><code>G</code></td>
<td>This inverts the &quot;greediness&quot; of the quantifiers so that they become greedy only when followed by a question mark.</td>
</tr>
<tr>
<td><code>I</code></td>
<td>Matches the unnormalized HTTP request URI buffer. This is similar to <code>http_raw_uri</code> in function. Snort does not allow using this modifier along with the HTTP request uri buffer modifier for the same content.</td>
</tr>
<tr>
<td><code>C</code></td>
<td>Matches normalized HTTP request or HTTP response cookie. This is similar to <code>http_cookie</code> in function. This is not allowed with the unnormalized HTTP request or HTTP response cookie modifier for the same content.</td>
</tr>
<tr>
<td><code>H</code></td>
<td>Matches normalized HTTP request or HTTP response header. This is similar to <code>http_header</code>. This modifier is not allowed with the unnormalized HTTP request or HTTP response header modifier for the same content. For SIP message, matches SIP header for request or response, similar to how <code>sip_header</code> functions.</td>
</tr>
</tbody>
</table>
## Modifier Description

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Matches unnormalized HTTP request body. This is similar to how <code>http_client_body</code> functions. For SIP message, matches SIP body for request or response, similar to how <code>sip_body</code> works.</td>
</tr>
<tr>
<td>D</td>
<td>Matches unnormalized HTTP request or HTTP response header, similar to <code>http_raw_header</code>. This modifier is not allowed with the normalized HTTP request or HTTP response header modifier for the same content.</td>
</tr>
<tr>
<td>M</td>
<td>Matches normalized HTTP request method, similar to <code>http_method</code>.</td>
</tr>
<tr>
<td>K</td>
<td>Matches unnormalized HTTP request or HTTP response cookie, similar to <code>http_raw_cookie</code>. This modifier is not allowed with the normalized HTTP request or HTTP response cookie modifier for the same content.</td>
</tr>
<tr>
<td>S</td>
<td>Matches HTTP response status code, similar to <code>http_stat_code</code>.</td>
</tr>
<tr>
<td>Y</td>
<td>Matches HTTP response status message, similar to <code>http_stat_msg</code>.</td>
</tr>
</tbody>
</table>

X, P, K, U, R, and C escape sequences and back references are not supported.

### Example:

```alert
tcp any any -> any any (msg: "command dot exe attempt"; content: "user"; pcre:"/cMd.eXe/i"; sid:2030; priority: 1;)
```

This rule first checks for the string, "user". Here, "user" is the triggering token. If "user" is found, the Sensor then checks for the PCRE "cmd.exe" in the first 256 bytes from the beginning of the buffer.

### byte_jump

This option is typically used to inspect length encoded protocols. When you use this option, the Sensor:

1. Goes to the specified location in the payload.
2. Reads the specified number of bytes.
3. Converts the number of bytes that was read to its decimal equivalent and jumps that much further into the payload from the current position.
To understand this option better, consider how the Sensor applies `byte_jump:2,6;` on a sample packet capture:

1. The value 6 in the example indicates that the Sensor should move its current position to the 7th byte from the beginning of the payload.

   \[\begin{array}{c}
   00 \quad 12 \quad 3f \quad 2c \quad 58 \quad 0b \quad 00 \quad 0b \quad bf \quad a1 \quad b7 \quad fc \quad 08 \quad 00 \quad 45 \quad 00 \\
   04 \quad 7f \quad 7b \quad ff \quad 00 \quad 00 \quad 32 \quad 0b \quad ab \quad 00 \quad 03 \quad 00 \quad 00 \quad 00 \quad 0c \\
   fc \quad e6 \quad a3 \quad 00 \quad 50 \quad 04 \quad 9c \quad 6b \quad b6 \quad 3e \quad 89 \quad 25 \quad 01 \quad 42 \quad 40 \quad 50 \quad 18 \\
   \ldots P..k>.%.B@P. \\
   26 \quad d8 \quad 39 \quad 1b \quad 00 \quad 00 \quad b0 \quad f1 \quad 1d \quad ca \quad 7c \quad dc \quad 54 \quad b6 \quad 5d \quad ac \\
   &.9.........]T.]. \\
   70 \quad 37 \quad 2b \quad 9a \quad t1 \quad d7 \quad 29 \quad 83 \quad 70 \quad 3e \quad d5 \quad c1 \quad 11 \quad a4 \quad e5 \quad e9 \\
   \ldots p7+...).p>........ \\
   6d \quad a0 \quad de \quad 9a \quad 9c \quad ae \quad f3 \quad 60 \quad 72 \quad 00 \quad 00 \quad 00 \quad 46 \quad e5 \quad 14 \quad e2 \\
   m........'-------- \\
   e8 \quad 97 \quad 53 \quad 49 \quad 42 \quad 88 \quad 68 \quad db \quad e2 \quad 45 \quad c4 \quad b2 \quad e8 \quad 46 \quad 49 \quad e1 \\
   \ldots SIB.h.E...FI. \\
   de \quad 8a \quad 81 \quad 4c \quad 33 \quad 31 \quad 6c \quad 49 \quad a1 \quad a2 \quad 5f \quad 2c \quad c6 \quad e4 \quad 50 \quad 90 \\
   \ldots L31I..._..P. \\
   9c \quad 94 \quad 3d \quad 31 \quad 70 \quad e4 \quad d9 \quad 42 \quad b0 \quad a7 \quad 85 \quad f8 \quad 02 \quad 4b \quad d0 \quad 3d \quad 2a \\
   \ldots =1p..B....+.=" \\
   02 \quad e4 \quad 67 \quad 4f \quad 04 \quad 1c \quad 79 \quad b6 \quad 08 \quad c5 \quad d3 \quad 22 \quad fc \quad 02 \quad 74 \quad 8f \\
   \ldots g0..y...."."t. \\
   8a \quad 70 \quad 7f \quad 7f \quad ef \quad f2 \quad ea \quad 66 \quad 2d \quad 41 \quad 80 \quad 7c \quad fc \quad 6b \quad 01 \quad 15 \\
   \ldots p....f-A.|.k.. \\
   a5 \quad 1c \quad 1c \quad 84 \quad e9 \quad f7 \quad 53 \quad 2e \quad 11 \quad e5 \quad a6 \quad 11 \quad 47 \quad 71 \quad f2 \quad e2 \quad ec \\
   \ldots S.....Gq... \\
   \end{array}\]

2. The value 2 indicates that the Sensor should read 2 bytes from the current position, which is 00 0b.

   \[\begin{array}{c}
   00 \quad 12 \quad 3f \quad 2c \quad 58 \quad 0b \quad 00 \quad 0b \quad bf \quad a1 \quad b7 \quad fc \quad 08 \quad 00 \quad 45 \quad 00 \\
   04 \quad 7f \quad 7b \quad ff \quad 00 \quad 00 \quad 32 \quad 0b \quad ab \quad 00 \quad 03 \quad 00 \quad 00 \quad 00 \quad 0c \\
   fc \quad e6 \quad a3 \quad 00 \quad 50 \quad 04 \quad 9c \quad 6b \quad b6 \quad 3e \quad 89 \quad 25 \quad 01 \quad 42 \quad 40 \quad 50 \quad 18 \\
   \ldots P..k>.%.B@P. \\
   26 \quad d8 \quad 39 \quad 1b \quad 00 \quad 00 \quad b0 \quad f1 \quad 1d \quad ca \quad 7c \quad dc \quad 54 \quad b6 \quad 5d \quad ac \\
   &.9.........]T.]. \\
   70 \quad 37 \quad 2b \quad 9a \quad t1 \quad d7 \quad 29 \quad 83 \quad 70 \quad 3e \quad d5 \quad c1 \quad 11 \quad a4 \quad e5 \quad e9 \\
   \ldots p7+...).p>........ \\
   6d \quad a0 \quad de \quad 9a \quad 9c \quad ae \quad f3 \quad 60 \quad 72 \quad 00 \quad 00 \quad 00 \quad 46 \quad e5 \quad 14 \quad e2 \\
   m........'-------- \\
   e8 \quad 97 \quad 53 \quad 49 \quad 42 \quad 88 \quad 68 \quad db \quad e2 \quad 45 \quad c4 \quad b2 \quad e8 \quad 46 \quad 49 \quad e1 \\
   \ldots SIB.h.E...FI. \\
   de \quad 8a \quad 81 \quad 4c \quad 33 \quad 31 \quad 6c \quad 49 \quad a1 \quad a2 \quad 5f \quad 2c \quad c6 \quad e4 \quad 50 \quad 90 \\
   \ldots L31I..._..P. \\
   9c \quad 94 \quad 3d \quad 31 \quad 70 \quad e4 \quad d9 \quad 42 \quad b0 \quad a7 \quad 85 \quad f8 \quad 02 \quad 4b \quad d0 \quad 3d \quad 2a \\
   \ldots =1p..B....+.=" \\
   02 \quad e4 \quad 67 \quad 4f \quad 04 \quad 1c \quad 79 \quad b6 \quad 08 \quad c5 \quad d3 \quad 22 \quad fc \quad 02 \quad 74 \quad 8f \\
   \ldots g0..y...."."t. \\
   8a \quad 70 \quad 7f \quad 7f \quad ef \quad f2 \quad ea \quad 66 \quad 2d \quad 41 \quad 80 \quad 7c \quad fc \quad 6b \quad 01 \quad 15 \\
   \ldots p....f-A.|.k.. \\
   a5 \quad 1c \quad 1c \quad 84 \quad e9 \quad f7 \quad 53 \quad 2e \quad 11 \quad e5 \quad a6 \quad 11 \quad 47 \quad 71 \quad f2 \quad e2 \quad ec \\
   \ldots S.....Gq... \\
   \end{array}\]
3. The Sensor converts the read bytes to decimal, which is 11.

4. The Sensor now skips 11 bytes from the current position as shown below:

```
00 12 3f 2c 58 0b 00 0b bf a1 b7 fc 08 00 45 00  ..?,X............E.
04 7f 7b ff 00 00 32 06 ab 00 03 00 00 00 0c  ..[...2.........
e6 a3 00 50 04 9c 6b b6 3e 89 25 01 42 40 50 18  ...P...k>.%.B@P.
26 d8 39 1b 00 00 b0 f1 1d ca 7c dc 54 b6 5d ac &.9.......[T].
70 37 2b 9a b1 d7 29 83 70 3e d5 c1 11 a4 e5 e9 p7+...).p>
6d a0 de 9a 9c ae f3 60 72 00 00 00 46 e5 14 e2 m.......`r.........
e8 97 53 49 42 88 68 db e2 45 c4 b2 e8 46 49 e1  ..SIB..h..E...FI.
de 8a 81 4c 33 31 6c 49 a1 a2 5f 2c c6 e4 50 90  ..L3I|I..._,...P.
9c 94 3d 31 70 e4 d9 42 b0 a7 85 f8 2b d0 3d 2a  ..=lp..B....+.=.k
02 e4 67 4f 04 1c 79 b6 08 c5 d3 22 fc 02 74 8f  ..g0..y...."..t.
8a 70 7f 7f ef f2 ea 66 2d 41 80 7c fc 8b 01 15  ..p......f-A.|.k.
a5 12 84 e9 f7 53 2e 11 e5 a6 11 47 71 f2 ec 16  .....S.....Gq...
```

The Sensor can also execute a byte_jump from a position relative to the previous match. Consider this example: content: "|06 ab 00 03|"; offset: 23; depth: 4; byte_jump: 2,3,relative.

1. The Sensor locates the content option as shown below.

```
00 12 3f 2c 58 0b 00 0b bf a1 b7 fc 08 00 45 00  ..?,X............E.
04 7f 7b ff 00 00 32 06 ab 00 03 00 00 00 0c  ..[...2.........
e6 a3 00 50 04 9c 6b b6 3e 89 25 01 42 40 50 18  ...P...k>.%.B@P.
26 d8 39 1b 00 00 b0 f1 1d ca 7c dc 54 b6 5d ac &.9.......[T].
70 37 2b 9a b1 d7 29 83 70 3e d5 c1 11 a4 e5 e9 p7+...).p>
6d a0 de 9a 9c ae f3 60 72 00 00 00 46 e5 14 e2 m.......`r.........
e8 97 53 49 42 88 68 db e2 45 c4 b2 e8 46 49 e1  ..SIB..h..E...FI.
de 8a 81 4c 33 31 6c 49 a1 a2 5f 2c c6 e4 50 90  ..L3I|I..._,...P.
9c 94 3d 31 70 e4 d9 42 b0 a7 85 f8 2b d0 3d 2a  ..=lp..B....+.=.k
02 e4 67 4f 04 1c 79 b6 08 c5 d3 22 fc 02 74 8f  ..g0..y...."..t.
8a 70 7f 7f ef f2 ea 66 2d 41 80 7c fc 8b 01 15  ..p......f-A.|.k.
a5 12 84 e9 f7 53 2e 11 e5 a6 11 47 71 f2 ec 16  .....S.....Gq...
```

2. Because you used the relative option, the Sensor skips 3 bytes from the last content match.
The Sensor reads 2 bytes, that is 00 0C. It then calculates the decimal equivalent, which is 12.

From the current position, that is, starting from e6, it skips 12 bytes to execute the next operation in the rule.

```
00 12 3f 2c 58 0b 00 0b bf a1 b7 fc 08 00 45 00 ..?,?,...........E.
04 7f 7b ff 00 00 32 06 ab 00 03 00 00 00 0c ..{....2.........
ed e6 a3 00 50 04 9c 6b b6 3e 89 25 01 42 40 50 18 ...P..k>.%.B@P.
26 d8 39 1b c0 00 b0 f1 1d ca 7c dc 54 b6 5d ac &.9............T..]
70 37 2b 9a b1 d7 29 83 70 3e d5 c1 11 a4 e5 e9 p7+...).p>.......6d a0 de 9a 9c ae f3 60 72 00 00 00 46 e5 14 c e2 m........`r.........
e8 97 53 49 42 68 68 db e2 45 c4 b2 e8 46 49 e1 ..SIB.h.E...FI.
dc 8a 81 4c 33 31 6c 49 a1 a2 5f 2c c6 e4 50 90 ...L317I.....,..P.
9c 94 3d 31 70 e4 d9 42 b0 a7 85 f8 2b d0 3d 2a ..=lP..B....+.=*
02 e4 67 4f 04 1c 79 b6 08 c5 d3 22 fc 02 74 8f ..g0..y......."..t.
8a 70 7f 7f cf f2 ea 66 2d 41 80 7c fc 6b 01 15 .p...f-A,.].k..
a5 12 84 e9 f7 53 2e 11 e5 a6 11 47 71 f2 ec 16 .....S.....Gq...
```

**Syntax:**
```
byte_jump:<bytes_to_convert>, <offset> [,, relative][, multiplier <mult_value>] [, <endian>][, string, <number_type>] [, align][, from_beginning][, post_offset <adjustment value>][, dce];
```

- **bytes:** 1 to 10
- **offset:** -65535 to 65535
- **mult_value:** 0 - 65535
- **post_offset:** -65535 to 65535

The following table describes the options that you can use with byte_jump:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>relative</td>
<td>See the example above.</td>
</tr>
<tr>
<td>multiplier &lt;integer&gt;</td>
<td>The Sensor multiplies the calculated decimal value with this integer and jumps that many bytes further into the payload.</td>
</tr>
<tr>
<td>big</td>
<td>Sensor processes the data as big endian. This is the default option.</td>
</tr>
<tr>
<td>little</td>
<td>Sensor processes the data as little endian.</td>
</tr>
<tr>
<td>string</td>
<td>To look for a string in the payload.</td>
</tr>
<tr>
<td>hex</td>
<td>Converted string is represented in hexadecimal.</td>
</tr>
<tr>
<td>dec</td>
<td>Converted string is represented in decimal.</td>
</tr>
<tr>
<td>oct</td>
<td>Converted string is represented in octal.</td>
</tr>
<tr>
<td>align</td>
<td>The number of converted bytes is rounded up to the next 32-bit boundary.</td>
</tr>
</tbody>
</table>
### Option  

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_beginning</td>
<td>Sensor executes the byte_jump from the beginning of the flow instead of</td>
</tr>
<tr>
<td></td>
<td>the current position.</td>
</tr>
<tr>
<td>post_offset &lt;positive</td>
<td>Sensor does an additional jump by the specified value (backward in case</td>
</tr>
<tr>
<td>or negative value&gt;</td>
<td>of negative value and forward in case of positive).</td>
</tr>
<tr>
<td>dce</td>
<td>A DCE/RPC request can specify whether numbers are represented in big</td>
</tr>
<tr>
<td></td>
<td>or little endian. Using this option in a rule facilitates the correct</td>
</tr>
<tr>
<td></td>
<td>conversion of the endianness of the request.</td>
</tr>
</tbody>
</table>

### byte_test

Like byte_jump, this option too is typically used to inspect length encoded protocols. For example, you can use this option to make sure a specific data in the payload does not exceed a certain length.

When you use this option, the Sensor:

1. Goes to the specified location in the payload
2. Reads the specified number of bytes
3. Converts the number of bytes that was read, to its decimal equivalent
4. Performs an operation on this decimal value against a value that you specify in the rule.

For example, you can use this option to make sure a specific data in the payload does not exceed a certain length.

**Syntax:**

```
byte_test: <bytes to convert>, [!]<operator>, <value>, <offset for byte_test> [,relative] [,<endian>] [, string, <number type>] [, dce];
```

- bytes to convert: 1 to 10
- operator: <, =, >, &, ^, <=, >=

For the description of these options, see the table for `byte_jump`.

**Example rule:**

```
alert tcp any any -> 10.1.1.1 80 (msg:"Example rule"; content:"|00 ab 00 00|"; flow:to_server; byte_test:5,>,200,18,relative; priority:1; sid:20089;rev:1;)
```

For this rule, the Sensor:

1. Locates the content in the payload.
2. If found, then from the end of the located string, it skips 18 bytes.
3. It reads 5 bytes and converts the binary value of the 5 bytes into decimal.
4. Checks if the converted value is greater than 200.

### byte_extract

This option is typically used to inspect length encoded protocols. For this option, the Sensor reads a specified number of bytes from the packet payload and stores it in a variable. It references these variables for subsequent processing. This option is supported only for Sensors on software version 8.0 or later.

Only two byte extract variables are allowed per rule. They can be re-used in the same rule any number of times.
The following are the rule options and their arguments that can use the byte_extract variables:

<table>
<thead>
<tr>
<th>Rule Option</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>content/uricontent</td>
<td>offset, depth, distance, within</td>
</tr>
<tr>
<td>byte_test</td>
<td>offset, value</td>
</tr>
<tr>
<td>byte_jump</td>
<td>offset</td>
</tr>
<tr>
<td>isdataat</td>
<td>offset</td>
</tr>
</tbody>
</table>

Syntax:

byte_extract: <bytes to extract>, <offset for byte_extract> <name of the variable> [,relative] [, multiplier <multiplier value>][, <endian>] [, string][, hex][, dec][, oct][, align <align value>][, dce];

The following table describes the options that you can use with byte_extract:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bytes to extract</td>
<td>Number of bytes to extract from the message.</td>
</tr>
<tr>
<td>offset</td>
<td>Number of bytes into the payload from where the Sensor should start processing.</td>
</tr>
<tr>
<td>name of the variable</td>
<td>Name of the variable, which other rule options in the rule will refer to.</td>
</tr>
<tr>
<td>relative</td>
<td>Refer to byte_jump.</td>
</tr>
<tr>
<td>multiplier &lt;integer&gt;</td>
<td>The Sensor multiplies the bytes read from the packet by &lt;integer&gt; and saves that number into the variable.</td>
</tr>
<tr>
<td>big</td>
<td>Sensor processes the data as big endian. This is the default option.</td>
</tr>
<tr>
<td>little</td>
<td>Sensor processes the data as little endian.</td>
</tr>
<tr>
<td>dce</td>
<td>A DCE/RPC request can specify whether numbers are represented in big or little endian. Using this option in a rule facilitates the correct conversion of the endianness of the request.</td>
</tr>
<tr>
<td>string</td>
<td>To look for a string in the payload.</td>
</tr>
<tr>
<td>hex</td>
<td>Converted string is represented in hexadecimal.</td>
</tr>
<tr>
<td>dec</td>
<td>Converted string is represented in decimal.</td>
</tr>
<tr>
<td>oct</td>
<td>Converted string is represented in octal.</td>
</tr>
<tr>
<td>align</td>
<td>The number of converted bytes is rounded up to the next 32-bit boundary.</td>
</tr>
</tbody>
</table>

Example rule:

alert tcp any any -> 10.1.1.1 80 (byte_extract:1, 0, str_offset; byte_extract:1, 1, str_depth; content: "example content"; offset: str_offset; depth:str_depth; msg:"An example for byte_extract"; priority: 1; sid:21089;rev:1;)

**ftpbounce**

Use this to detect the FTP bounce attack. However, the rule converts with a warning and is not included in the policies when you save it in the Manager server because an equivalent McAfee attack is available.

Syntax: ftpbounce;
This option is not relevant for a Snort rule in Network Security Platform, and is not supported. If a rule that you imported contained this option, it is displayed in the Conversion Notes section of the rule. The Sensor ignores this option when it checks the traffic against this rule.

**cvs**

This option is not relevant for a Snort rule in Network Security Platform, and is not supported. If a rule that you imported contained this option, it is displayed in the Conversion Notes section of the rule. The Sensor ignores this option when it checks the traffic against this rule.

**dce_iface**

This rule option checks for the specified interface UUID of the DCE/RPC request. Optionally, you can also use it to check the version operation.

Syntax:

```
dce_iface:<uuid>[, <operator><version>][, any_frag];
```

- The uuid must be in the following format: 4hexbyte - 2hexbyte - 2hexbyte - 2hexbyte - 6hexbyte. For example: 123ef678-1ac4-a2cd-ef01-0a23bc6789cb
- Supported operators are, <, >, =, !
- Version can be an integer between 0 and 65535.
- By default, the Sensor applies the rule on a first fragment (or full request, that is, not a fragment). To apply the rule on the middle and last fragments as well, use the any frag argument. Note that a defragmented DCE/RPC request is considered a full request.

Examples:

- `dce_iface:4b324fc8-1670-01d3-1278-5a47bf6ee188;` — This matches if the specified interface UUID matches the interface UUID and if the fragment is a first fragment or full request.
- `dce_iface:12345678-1234-abcd-ef00-0123456789ab, <2;` — This matches if all of the following are met:
  - The specified interface UUID matches the interface UUID.
  - The version is less than 2.
  - The fragment is a first fragment or full request.
- `dce_iface:12345678-1234-abcd-ef00-0123456789ab, any_frag;` — This matches if the specified interface UUID matches the interface UUID.
- `dce_iface:12345678-1234-abcd-ef00-0123456789ab, =1, any_frag;` — This matches if the specified interface UUID matches and the version is equal to 1.

Example rule:

```
alert tcp any any -> 10.1.1.1 445 (msg:"Example rule"; flow:to_server; dce_iface: 12345678-1234-abcd-ef00-0123456789ab; priority:1; sid:20489;rev:1;)
```

**dce_opnum**

You use this option to check for specific operation numbers (opnum) of a DCE/RPC request. The opnum indicates the specific function call to an interface. It enables you to know the function call a client is making to a service after you establish that the client is bound to a specific interface and is making a request to it.
Syntax:

dce_opnum:opnum-item | opnum-range | opnum-list where opnum is an integer between 0 and 65535.

Examples:

- dce_opnum:10;
- dce_opnum:10-15;
- dce_opnum:10, 20-25;
- dce_opnum:10, 12, 20-25

Example rule:

alert tcp any any -> 10.1.1.1 445 (msg:"Example rule"; flow:to_server; dce_iface: 12345678-1234-abcd-ef00-0123456789ab; dce_opnum:7; priority:1; sid:20689;rev:1;)

dce_stub_data

This option makes the Sensor to point to the beginning of the DCE/RPC stub data.

Syntax: dce_stub_data;

Example rule:

alert tcp any any -> 10.1.1.1 445 (msg:"Example rule"; flow:to_server; dce_iface: 12345678-1234-abcd-ef00-0123456789ab; dce_opnum:7; dce_stub_data; content:"|00 00 00 00|"; depth:4; offset:8; priority:1; sid:20689;rev:1;)

sip_method

This rule option enables you to check for Session Initiation Protocol (SIP) request methods. In the same option, you can specify multiple SIP request methods separated by commas. In this case, it is considered as an OR condition. That is, the rule triggers if there is a match for any of the mentioned request methods.

Syntax:

sip_method:<method>|<method-list>

The following request methods are supported:

- invite
- cancel
- ack
- bye
- register
- options
- refer
- subscribe
- update
- join
- info
- message
- notify
- prack

The ! operator is supported. However, if you use !, you can specify only one method in one sip_method option.
Examples:

- `sip_method:invite` — This checks for the invite request method.
- `sip_method:!invite` — The condition is true if the method is anything other than invite.
- `sip_method:invite,cancel,bye` — The condition is true if the method is invite, cancel, or bye.
- `sip_method:!invite; sip_method:!cancel` — The condition is true if the method is anything other than invite and cancel.

Example rule:

```
alert udp any any -> 10.1.1.1 5060 (msg:"Example rule"; flow:to_server; sip_method:invite; content:"SIP/2.0"; nocase; priority:1; sid:20189;rev:1;)
```

**sip_stat_code**

This rule option enables you to check for SIP response status codes. The condition matches if any of the specified status code is present in the SIP response.

Syntax:

```
sip_stat_code:<code>|<code>,<code>
```

The following request methods are supported:

Examples:

- `sip_stat_code:400` — This checks for the status code 400.
- `sip_stat_code:180,182` — The condition matches if the status code is 180 or 182.
- `sip_stat_code:4` — This condition looks for 4xx. That is it is true if the method is anything from 400 to 599. The numbers 1 to 6 are expressed as 1xx, 2xx, 3xx, and so on where, for example, 1xx corresponds to the range 100 - 199.

Example rule:

```
alert udp any any -> 10.1.1.1 5060 (msg:"Example rule"; flow:to_server; sip_stat_code:401; content:"SIP/2.0"; nocase; priority:1; sid:20289;rev:1;)
```

**sip_header**

This rule option searches only the extracted header fields of a SIP request or response.

Syntax: `sip_header;`

Example rule:

```
alert udp any any -> 10.1.1.1 5060 (msg:"Example rule"; flow:to_server; sip_header; content:"SIP/2.0"; nocase; priority:1; sid:20290;rev:1;)
```

**sip_body**

This rule option points the Sensor to the beginning of the body fields of a SIP message.

Syntax: `sip_body;`

Example rule:

```
alert udp any any -> 10.1.1.1 5060 (msg:"Example rule"; flow:to_server; sip_body; content:"SIP/2.0"; nocase; priority:1; sid:20491;rev:1;)
```
**ssl_version**

This rule option can check for the SSL version numbers exchanged between the server and the client during the handshake. In the same option, you can check for multiple SSL versions separated by commas. In this case, it is considered as an OR condition. That is, the rule triggers if there is a match for any of the versions. To check for an AND condition involving two or more versions, use separate `ssl_version` rule options.

Syntax:

```
ssl_version:<version-list>
```

The `!` operator is supported. However, if you use `!`, you can specify only one method in one `sip_method` option.

Examples:

- `ssl_version:sslv2` — This checks for SSL version 2.
- `ssl_version:!sslv3` — The condition is true for any version other than SSL version 3.
- `ssl_version:tls1.0,tls1.1,tls1.2` — The condition is true if the SSL version is TLS 1.0, TLS 1.1, or TLS 1.2.
- `ssl_version:!tls1.0; ssl_version:!tls1.1` — The condition is true if the SSL version is anything other than TLS 1.0 and TLS 1.1.

Example rule:

```
alert tcp any any -> 10.1.1.2 443 (msg:"Example rule"; flow:to_server; ssl_version:sslv2; content:"|0B|"; priority:1; sid:20389;rev:1;)
```

**ssl_state**

This option is not relevant for a Snort rule in Network Security Platform, and is not supported. If a rule that you imported contained this option, it is displayed in the Conversion Notes section of the rule. The Sensor ignores this option when it checks the traffic against this rule.

**modbus_func**

This rule option checks for the specified function code in the in a Modbus header. You can specify either the code number in the decimal format or the equivalent string.

Syntax: `modbus_func:<code>`

The following are the supported values for code:

- A number ranging from 0 to 255
- `read_coils`
- `read_discrete_inputs`
- `read_holding_registers`
- `read_input_registers`
- `write_single_coil`
- `write_single_register`
- `read_exception_status`
- `get_comm_event_log`
- `write_multiple_coils`
- `write_multiple_registers`
- `report_slave_id`
- `read_file_record`
- `write_file_record`
- `mask_write_register`
- `read_write_multiple_registers`
• diagnostics
• get_comm_event_counter

Examples:
• modbus_func:1
• modbus_func:read_discrete_inputs

Example rule:
alert tcp any any -> 10.1.1.5 502 (msg:"Example rule"; flow:to_server; modbus_func:write_multiple_coils; byte_test:2,>\,1968,10; reference:url,www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf; classtype:protocol-command-decode; priority:1; sid:20589;rev:1;)

**modbus_unit**
This rule option checks against the unit ID field in a Modbus header.

**Syntax:** modbus_unit:<unit>; where unit can be a value from 0 to 255.

**Example:** modbus_unit:1;

Example rule:
alert tcp any any -> 10.1.1.5 502 (msg:"Example rule"; flow:to_server; modbus_unit:1; byte_test:2,>\,1968,10; reference:url,www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf; classtype:protocol-command-decode; priority:1; sid:20590;rev:1;)

**modbus_data**
This rule option points the Sensor to the beginning of the data field in a Modbus request or response.

**Syntax:** modbus_data;

Example rule: alert tcp any any -> 10.1.1.7 502 (msg:"Example rule"; flow:to_server; modbus_data; content:"example content"; nocase; priority:1; sid:20481;rev:1;)

**dnp3_func**
This rule option checks against the function code of a DNP3 request or response header. You can specify either the code number in the decimal format or the equivalent string.

**Syntax:**
dnp3_func:<code>

The following are the supported values for code:

- A number ranging from 0 to 255
- stop_appl
- confirm
- save_config
- read
- enable_unsolicited
- write
- disable_unsolicited
- select
- assign_class
- operate
delay_measure
Mechanics of a Snort custom attack
Structure of a snort rule

- direct_operate
- direct_operate_nr
- immed_freeze
- immed_freeze_nr
- freeze_clear
- freeze_clear_nr
- freeze_at_time
- freeze_at_time_nr
- cold_restart
- warm_restart
- initialize_data
- initialize_appl
- start_appl

Examples:
- dnp3_func:2;
- dnp3_func:get_file_info;

**dnp3_ind**

This rule option checks the internal indicators flags present in a DNP3 Application response header. If you specify multiple flags in the same dnp3_ind option, it results in an OR condition. For an AND condition, use multiple dnp3_ind options.

Syntax:

```
dnp3_ind:<flag>{,<flag>...}
```

The supported flags are:

- all_stations
- class_1_events
- class_2_events
- class_3_events
- need_time
- local_control
- defice_trouble
- device_restart
- no_func_code_support
- object_unknown
- parameter_error
- event_buffer_overflow
- already_executing
- config_corrupt
- reserved_2
- reserved_1
Examples:

- `dnp3_ind:device_trouble, device_restart` — This matches if the flag is `device_trouble` or `device_restart`.
- `dnp3_ind:class_1_events; dnp3_ind:class_2_events` — This matches for `class_1_events` and `class_2_events`.

**dnp3_obj**

This rule option checks on the DNP3 object headers in a request or response.

Syntax:

```
dnp3_obj:<group>,<var>
```

Group and var can be integer values between 0 and 255.

Example:

```
dnp3_obj:50,1; — Matches for the DNP3 Date and Time object.
```

**dnp3_data**

This rule option points the Sensor to the beginning of the application-layer fragment so that the Sensor can execute the other rule options.

Syntax:

```
dnp3_data;
```

Example:

```
dnp3_data; content:"example content";
```

**Non-payload rule options**

This section explains Snort non-payload rule options.

**id**

Using this option, you can specify an IP identification value that you want the Sensor to look for in the IP header. It is possible to identify some attacks by the IP identification value. Note that you can specify only a decimal `id:<number>;al value`.

Syntax: `id:<number>;`;

Example rule: `alert ip any any -> any any (msg:"example for IP ID"; id:31238; content:"red"; sid:2011; priority:1;)`

**fragbits**

You use this option to check if the following fragmentation and reserved bits are set in the IP header.

- **M** — More fragments
- **D** — Don't fragment
- **R** — Reserved bit
Set the following modifiers to change the match criteria:

- + to match on the specified bits and any others
- * to match if any of the specified bits are set
- ! to match if the specified bits are not set

Syntax: \texttt{fragbits: [+!*!]<[MDR]>;}

Example rule: alert ip any any -\rightarrow any any (msg:"example for fragbits"; fragbits:MD; sid:2023; priority:1;)

**fragoffset**

Using this keyword you can compare the IP fragment offset field against a decimal value. To identify all the first fragments of an IP session, you can use the fragbits keyword and look for the More fragments option in conjunction with a fragoffset of 0.

Syntax: \texttt{fragoffset: [+|-]<number> ;}

Example rule: alert ip any any -\rightarrow any any (msg:"example for fragoffset"; fragbits:M; fragoffset:0; sid:2123; priority:1;)

**ttl**

You can use this to check the time-to-live (TTL) value in the IP header. Abnormally low or high TTL may indicate misuse of network resources.

Syntax: \texttt{ttl: [<=]<number> ;}

Example rule: alert tcp any any -\rightarrow any any (msg:"example for ttl"; ttl:<5; sid:2013; priority:1;)

This rule triggers for ttl less than 5.

**tos**

Use this if you want the Sensor to check on the TOS value in an IP header.

Syntax: \texttt{tos: [+]<number> ;}

Example rule: alert tcp 10.1.1.1 any -\rightarrow any any (msg:"example for tos"; tos:7; sid:2014; priority:1;)

This rule triggers if traffic from 10.1.1.1 has a tos of 7.

**ipopts**

Use this option if you want the Sensor to check for the following IP options:

- rr - Record Route
- eol - End of list
- nop - No Op
- ts - Time Stamp
- sec - IP Security
- esec - IP Extended Security
- lsrr - Loose Source Routing
• ssrr - Strict Source Routing
• satid - Stream identifier
• any - any IP options are set

You can specify ipopts only once in a rule.

Syntax

ipopts:<rr|eol|nop|ts|sec|esec|lsrr|ssrr|satid|any>;

Example rule: alert ip any any -> any any (msg:"example for ipopts"; ipopts:ssrr; sid:2012; priority: 1;)

dsize

Use this if you want the Sensor to check the size of the payload. That is, payload size excluding the TCP, UDP, or ICMP header. Typically, you use this to check packets of abnormal sizes such as in case of buffer overflows.

dsizer will fail on stream rebuilt packets, regardless of the size of the payload.

Syntax: dsize:[<>]<number>[<>]<number>;

Example rule: alert ip any any -> any any (msg:"example for dsize"; dsize:1000<>1500; sid:2016; priority:1;)

This checks for packets with a payload between 1000 and 1500 bytes in size.

ip_proto

Use this option to check the IP header for the next level protocol. You can specify the protocol name or number. For example, you can specify tcp or 6 to check for TCP traffic.

Syntax: ip_proto:![>|<] <name or number>;

Example rule: alert ip any any -> any any (msg:"example for ip protocol"; ip_proto:icmp; priority: 3;sid:200709;rev:2;)

This rule checks the IP header if the next level protocol is ICMP.

sameip

This keyword is not supported in Network Security Platform. If you import a rule with this keyword, it imports with a warning, but the Sensor ignores the rule. However, there is an equivalent attack definition in McAfee's signature set. So, the Sensor raises an alert for this traffic based on McAfee's attack definition.

flags

Use this if you want the Sensor to check for TCP flags. You can check for the following flags:

• F - FIN (LSB in TCP Flags byte)
• U - URG
• S - SYN
• 1 - Reserved bit 1 (MSB in TCP Flags byte)
• R - RST
• 2 - Reserved bit 2
• P - PSH
• 0 - No TCP Flags Set
• A - ACK

You can apply the following operators on the flags option:

• + - check if all the specified flags are set
• * - check if at least one of the specified flags are set
• ! - check if none of the flags are set

Syntax: flags:[!|*|+]<FSRPAU120>[,<FSRPAU120>];

Example rule: alert tcp 10.1.1.1 any -> any any (msg:"example for flags"; flags:R; sid:2019; priority: 3;)

This rule raises an alert when the host 10.1.1.1 resets a TCP connection.

**flow**

This option enables you to write rules specific to a direction of the traffic flow. For example, you can write a rule that the Sensor applies only on the traffic from the clients.

The following table describes the flow options that you can use in Network Security Platform:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_client</td>
<td>Rule is applicable only to the response traffic from the server in a TCP session.</td>
</tr>
<tr>
<td>to_server</td>
<td>Rule is applicable only to the request traffic from the client in a TCP session.</td>
</tr>
<tr>
<td>from_client</td>
<td>Same as to_server</td>
</tr>
<tr>
<td>from_server</td>
<td>Same as to_client</td>
</tr>
</tbody>
</table>

Syntax: flow: [, (to_client|to_server|from_client|from_server)];

Example rule: alert tcp !$HOME_NET any -> 10.1.1.1 80 (msg:"example for flow"; flow:to_server; content:"cmd.exe";sid:2015; priority:1;)

For this rule, the Sensor checks for cmd.exe in the request traffic from outside network to 10.1.1.1.

**Notes:**

• Make sure you specify the flow option in the Snort custom attacks to avoid false-positives.
• If you do not specify flow for a TCP rule, then it is saved in the Manager with Conversion Result, "warning" and State, "Included."

**flowbits**

Use this option to keep track of an application protocol across flows or even sessions. You use this typically for TCP sessions. You create a rule in which you specify the Sensor to set or unset a state when the rule returns positive. You need to specify the name for this state. Then you create another rule and base it on the same state. See the example below.

alert tcp any any -> any 80 (msg:"example rule 1 for flowbits"; content:"abc"; flowbits:set,abc_found;flowbits:noalert; sid:2035; priority:1;)

alert tcp any any -> any 80 (msg:"example rule 2 for flowbits"; content:"def"; flowbits:isset,abc_found;sid:2036; priority:1;)

The first rule checks for "abc". If found, the Sensor sets the state "abc_found" but raises no alert. The second rule checks for "def" in the same flow as the first, and if found the Sensor raises an alert.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set</td>
<td>Sensor sets the specified state for the current flow.</td>
</tr>
<tr>
<td>unset</td>
<td>Clears the specified state for the current flow.</td>
</tr>
<tr>
<td>toggle</td>
<td>Sets the state if unset; clears it if already set.</td>
</tr>
<tr>
<td>isset</td>
<td>Checks if the state is set.</td>
</tr>
<tr>
<td>isnotset</td>
<td>Checks if the state is not set.</td>
</tr>
<tr>
<td>noalert</td>
<td>Sensor does not raise an alert for this rule.</td>
</tr>
</tbody>
</table>

**sseq**

Use this if you want the Sensor to look for a specific TCP sequence number.

Syntax: `seq:<number>;`

**ack**

Use this if you want the Sensor to look for a specific TCP acknowledgement number.

Syntax: `ack: <number>;`

Example rule: alert tcp !$HOME_NET any -> $HOME_NET 80 (msg:"example for seq and ack numbers"; seq:0;ack:0;sid:2020; priority:2;)

This rule is triggered when both sequence and acknowledgement numbers are 0 when the traffic is flowing from outside to the inside network.

**window**

Use this if you want the Sensor to look for a specific TCP window size.

Syntax: `window:![<|>]<number>;`

Example rule: alert tcp !$HOME_NET any -> 11.1.1.10 80 (msg:"example for window size"; window: 360;sid:2022; priority:3;)

This rule is triggered when the TCP window size for host 11.1.1.10 is 360 bytes.

**stream_size**

This enables you to match traffic according to the number of bytes observed, as determined by the TCP sequence numbers.

Syntax: `stream_size:<server|client|both|either>, <operator>, <number>;`

The operator can be one of the following:

- `<`
- `!=`
- `>`
- `<=`
- `=#`
- `>=`

Example rule: alert tcp any any -> any any (msg:"example for stream_size"; stream_size:either,>,12800; sid:2032; priority:3;)

**itype**

Use this if the Sensor is to check for a specific ICMP type value.

Syntax: `itype: [<|>]<number>[<|><number>];`
**icode**

Use this if the Sensor is to check for a specific ICMP code value.

**Syntax:** icode: [<>][<number>][<>][<number>];

**Example rule:** alert icmp !$HOME_NET any -> $HOME_NET any (msg:"example for itype and icode"; itype:8; icode:0; sid:2022; priority:3;)

This rule is triggered for inbound traffic where the type is 8 and code is 0.

**icmp_id**

Use this to check for a specific ICMP ID value. You can prevent the use of static ICMP fields by malicious programs in their communication.

**Syntax:** icmp_id:<number>;

**Example rule:** alert icmp !$HOME_NET any -> $HOME_NET any (msg:"example for icmp_id"; icmp_id:0; sid:2040; priority:3;)

**icmp_seq**

This is to check for a specific ICMP sequence value.

**Syntax:** icmp_seq:<number>;

**Example rule:** alert icmp !$HOME_NET any -> $HOME_NET any (msg:"example for icmp_seq"; icmp_seq:0; sid:2041; priority:3;)

**rpc**

This keyword is not supported in Network Security Platform. If you import a rule with this keyword, it imports with a warning, but the Sensor ignores the rule.

**Post-detection rule options**

Post-detection rule options are irrelevant for Snort Custom Attacks. After an attack is saved in the Manager server, you can configure the response actions for the attack just like you would for any other attack definition in Network Security Platform.
Management of snort custom attacks

This section provides information on how to create and maintain Custom Attacks using Snort rules syntax. In Network Security Platform, you can create Snort Custom Attacks in one of two ways:

- You can construct the Snort rules, one at a time, directly in the Custom Attack Editor.
- You can construct multiple attacks in a file and import the file into the Manager. You may want to consider this method when you have a large number of attack definitions to create or if you want to use the Snort rules from a source such as the Snort user community.

Before you use Snort Custom Attacks, make sure you have reviewed and understood the considerations and best practices to be followed when using Snort rules in Network Security Platform.

See also

Management of custom attacks from the Central Manager on page 5

Contents

- Considerations
- Best practices
- Create snort custom attacks
- Variables
- Define the snort variables
- Define the classification types
- View the variables and classification types
- Identification of the protocol of a snort custom attack
- How to use snort rules to detect IP communication between specific hosts
- Importing snort custom attacks
- Write snort custom attacks
- Saving the Snort custom attacks
- Save the qualified rules
- Customizing the snort rules attack responses
- Delete the snort rules from the database

Considerations

Review the following very critically if you plan to use Snort Custom Attacks:

- An I-series Sensor can support up to 2500 Snort Custom Attacks, M-series and NS-series up to 5000. If you require more Snort Custom Attacks to be supported, contact McAfee Support.
- \X, \P, \K, \U, \R, and \C escape sequences and back references are not supported.
- Isdataat with rawbytes modifier is not supported.
- Norm and raw modifiers are not supported in urilen.
- Snort rules cannot be created on ASN1 decoded content.
- If the length of the content or uricontent (token length) is more than 96 bytes and lesser than or equal to 256 bytes, then the target device is set to M-series and NS-series.
- The following PCRE constructs are not supported:
  - Lookahead and lookbehind assertions.
  - Backreferences and capturing subexpressions.
- The following are not supported in Snort Custom Attacks, but there is an equivalent option in Network Security Platform:
  - Rules using ftpbounce keyword are not supported. However, the McAfee signature sets provide adequate protection against FTP bounce attacks.
  - Rules to detect CVS attacks are not supported. However, McAfee signature sets can protect your network against most of the CVS attacks.
• Regarding Sensor response actions, you cannot use any post-detection rule options such as Logto, Session, or Tag. Instead, you can configure the required response action after the rule is saved in the Manager database. This is the same as how you would configure the response action for any other attack definition in Network Security Platform.

• Preprocessor plugins are not supported.

• Snort’s multi-event logging (event queue) is not supported.

• For event thresholding and alert suppression, you can use only the equivalent features in Network Security Platform and not the options in Snort.

Important note regarding Snort custom attack signatures

Beginning with version 8.0, Snort custom attacks are translated into a newer McAfee signature format. This is required to support more Snort rule options as well as for performance improvement. However, the Snort custom attacks in this newer format are incompatible with 7.x Sensors. So, to support a heterogeneous Sensor environment, 2 signatures are created for each Snort custom attack - one for 8.0 Sensors and the other for 7.x Sensors.

After you upgrade the Manager from 7.x to 8.0 or later, it is mandatory that you re-submit all the Snort custom attacks for translation. Then, two signatures are created for those rules as well. To re-submit the rules, in the Custom Attack Editor, select File | Snort Advanced | View Snort Variables | Re-Submit Rules using Current Variables.

The two signatures are created regardless of whether you have a 7.x Sensor in your setup. This is to address scenarios where one might add a 7.x Sensor to an 8.0 Manager at a later time. If you do not require the signature for 7.x Sensors, you can turn it off.

1. Locate the ems.properties file. On the Manager server, go to \App \config\.

2. In the ems.properties file, un-comment # iv.snortimport.translation.tpuverion.0.support=false. That is, change this line to iv.snortimport.translation.tpuverion.0.support=false.

3. Re-start the Manager service.
Additional considerations for I-series Sensors

- If you use PCRE option in a rule, then there must be a Content keyword preceding the PCRE option. This Content keyword is the triggering token for the PCRE option. That is, the I-series Sensor considers the PCRE option only if the traffic returns positive for the preceding Content. Do not use negation in the triggering token.
- The triggering token content and the PCRE option in a rule must refer to the same modifier to be considered for I-series Sensors.
- For PCRE options, an I-series Sensor scans only up to 256 bytes from the beginning of the triggering token.

Best practices

This section details the best practices that you must follow when you use Snort Custom Attacks:

- Do not use a Snort Custom Attack if there is an equivalent available in the signature set.
- Make sure that the content option value is more than one byte. If you import a rule with a one-byte content, it will fail to import. The longer the content value, the accurate the detection will be. However, the maximum length of a content or uricontent for I-series Sensors is 96 bytes and 256 bytes in case of M-series and NS-series.
- Make sure the content option does not contain any generic values identified by McAfee (some examples are listed below). Such rules can severely impact Sensor performance.
  - GET
  - POST
  - Host:
  - User-Agent
- For better accuracy and performance, McAfee recommends that you use the Custom Attack Editor to create custom attack definitions as opposed to importing Snort rules.
- If you are using byte_test or byte_jump, use them in relation to a content match.
- Specify the classtype or priority to all rules. This enables the Manager to determine the severity for the rule. Understand how the Manager categorizes a Snort Custom Attack to include it in the rule sets.
- If you are importing the Snort rules, then import them from files that are accordingly named. For example, import http rules from a file named http.rules file. In these rules, do not specify the destination port; the Sensor automatically detects protocols running on non-standard ports and applies the rule to the corresponding traffic. If you specify a port number, then the Sensor applies the rule only to the traffic destined for that port.
- If you create the Snort rule in the Custom Attack Editor, or if you import it from a generically named file (like myrules.rules), then it is very important that you specify the destination port number.
- Specify the revision number for all rules.
- For TCP rules, specify the flow.
- In a rule, do not specify the same value for more than one Content option. For example, do not use a Snort Custom Attack such as this: alert tcp any any -> 10.1.1.1 80 (msg:"Example rule"; content:"private"; content:"private"; priority:1;sid:20209;rev:1;).
Create snort custom attacks

The following is the high-level approach for creating Snort Custom Attacks.

Task
1. Have a clear understanding on what you are trying to achieve through these attacks. See if there are any alternatives or better methods to achieve the same. Identify the related protocols, applications, hardware and software platforms, and so on for the attack. Make sure you have all the required information at hand.

2. Understand the mechanics of a Snort Custom Attack. You need to have a strong understanding of network concepts as well as Snort rules language to create effective Snort Custom Attacks.

3. Take a quick tour of the Custom Attack Editor. Get familiar with the Custom Attack Editor - the tool that you will use to manage Custom Attacks. Understand the interfaces related to Snort Custom Attacks.

4. Verify the required variables. Check if the variables that you plan to use are available in the Sensor.

5. Create Snort Custom Attacks. There are two methods:
   - You can import the rules from a file.
   - You can construct the Snort rules directly in the Custom Attack Editor.

6. Check the rules that failed to convert. The Sensor automatically converts all the valid rules, which you imported or wrote, to McAfee’s proprietary format. Note that some rules could have failed to convert. Troubleshoot and fix the rules that failed to convert. After you fix the rules, make sure you use the Test Compile feature on them to verify their validity and compatibility.

7. Save the converted rules in the Sensor database.

8. Once saved in the database, the rules are like any other custom attack definition. For example, you may want to customize the response action for the saved Snort rules or you may want to delete a Snort rule from the database.

9. Update the respective Sensors with the changes.
   Now, the Sensors raise alerts based on the saved Snort rules as well. You can view these alerts in the Threat Analyzer.

Variables

There can be some values that you may need to mention in all or most of the rules. For example, the target subnet could be the same for many rules. Instead of repeating it in each rule, you can define a variable for the subnet, and then use the variable in the rules. The advantage with using variables is when you want to modify the value. For example, if the target subnet is now different, then instead of modifying each rule, you can just redefine the variable with the new value.
Define the snort variables

You can define the snort variables in Network Security Platform and then use it in the rules.

**Task**
1. Define the variables with the appropriate values in a file.
2. Import the file into the Sensor.

   To define the variables in a file:
   a. Create a text file and change its file extension to .rules or .conf.
      Assume that you have named it variables.conf
   b. In the text file, define the variables as explained below
      - Use the `var` keyword to define a variable for a file path, IP addresses, and ports.
        For example, `var RULE_PATH ../rules`.
        In this example, RULE_PATH is the variable name and its value is the relative path to a folder named "rules".
      - Use the `ipvar` keyword to define a variable for IP addresses. Some examples are:
        - `ipvar INSIDE_NETWORK [10.1.1.0/24, !10.1.1.22, 11.1.1.1, 12.1.1.0/24]`
        - `ipvar EXAMPLE1 [$INSIDE_NETWORK, !10.1.1.23]`
        - `ipvar EXAMPLE2 [$EXAMPLE1]`
        - `ipvar EXAMPLE3 [1.1.1.1, 2.2.2.0/24, ![2.2.2.2, 2.2.2.3]]`
      - Use the `portvar` keyword to define the port numbers
        For example, `portvar EXAMPLE_PORTS [100, 102, 150:160, !155]`
        In this example, the value of EXAMPLE_PORTS is 100, 102, 150 through 160 except 155.
   c. Save the variables.conf file.
To import the variables file into Sensor:

a. In the Custom Attack Editor, select File | Import | Snort Rules.

b. Locate variables.conf and click Open.

   If you do not see the conf file at the location where you saved, check the Files of Type field in the Open dialog.

c. Click Open.

   The Import Status may show zero for all the fields. Click OK.

d. Select File | Snort Advanced | View Snort Variables to make sure the variables are imported with the values you specified in the variables.conf file.

e. Click File | Save to save the imported variables to the database.

f. If a variable that you imported is already available in the database, it is assigned the value from the current import.

---

**Define the classification types**

It is mandatory that you define the priority for each Snort Custom Attack. Priority is one of the parameters that the Manager uses when it categorizes a Snort Custom Attack. To specify the priority, you can either use the priority keyword directly in the rule or use a classification type in the rule. However, this classification type should either be available in the Manager or declared in a file up front. That is, the Manager must read the classification type declaration before it reads the rule that uses it.

You can view the classification types that are currently available in the Manager. You can also re-submit rules for conversion with the currently available values.

Similar to variables, you can define the classification types in a .conf, .config, or .rules file and import it into the Manager.

In the .conf, .config, or .rules file, define the classification type using the syntax as shown below:

```
config classification:<class name>,<class description>,<default priority>
```

Example: `config classification: brute-force,attempted brute force,1`

Note the following:

- The priority value 1 is mapped to a severity of high in Network Security Platform, 2 to medium, 3 to low, and 4 and above to informational.

- The default priority value that you specify in a classification type can be overridden by using the priority keyword in the rule.

**See also**

*Import snort rules through a conf file on page 119*
*Import snort rules through a rules file on page 121*
*Create a snort custom attack — an example on page 145*

---

**View the variables and classification types**

You can use the Snort Advanced feature to view the names and values of the variables, classification types, and references that are currently available in the Manager database.
This feature enables you to:

- check if you are using valid variables and classification types when you create or import a Snort Custom Attack. You cannot create a Snort Custom Attack that contains an undefined variable or classification type. If you import a rule with an undefined variable or classification type, it fails to get converted to McAfee’s format.

- verify if a variable or classification type that you want to define is already available.

- verify if an import of variables or classification types was successful.

To view the names of the variables, classification types, and references in the Manager:

**Task**

- In the Custom Attack Editor, select File | Snort Advanced | View Snort Variables.

The Snort Variables dialog opens. In this dialog, the variables and the values are listed in the Macros section. The classifications and references are listed in the corresponding sections.

![Snort Variables window](image)

**Figure 7-1  Snort Variables window**

**Re-submitting rules with the current variables and classification types:**

In the Snort Variables dialog, click **Re-submit Rules using Current Variables** if you want the Manager to re-convert all the rules in the All Custom Attacks tab to McAfee’s format but using the currently available variables.

This feature is useful if you had:

- imported the rules before you defined the variables. If you re-submit the rules now, the ones that failed to convert because of invalid variables get successfully converted. However, note that all the rules listed in the All Custom Attacks tab are submitted for re-conversion and not just the failed ones.

- imported the variables with modified values and you want the Manager to re-convert the rules with the new values.
Deleting variables and classification types from the Manager

To delete all the listed variables, classification types, and references from the Manager:

1. Click Delete Current Variables.
   This deletes the details from the Manager client.

2. Select File | Save.
   This deletes the details from the Manager server.

You cannot delete a specific variable, classification type, or reference. When you click Delete Current Variables, the entire list of variables, classification types, and references are deleted from the Manager client.

Identification of the protocol of a snort custom attack

When you save a custom attack in the Manager server, the Manager categorizes the attack to include it in the applicable rule sets. One of the criterion that the Manager uses to categorize an attack is the application layer protocol that the attack is intended for. For McAfee Custom Attacks, you can specify this when creating the attack. For Snort Custom Attacks, the Manager identifies it by itself.

Understand how the Manager identifies the impact protocol for a Snort Custom Attack, to ensure if a Snort Custom Attack is included in the policies it is intended for.

How the Manager identifies the impact protocol is explained below:

1. First, it tries to use the name of the rules file from which you imported the Snort Custom Attack. For example, if you imported it from ftp.rules, then the impact protocol for that attack is identified as FTP. Depending on other criteria for classification, the Manager includes the attack definition in the Rule Sets that include ftp.

   If the rules file name starts with web, for example web-attacks.rules or web-client.rules, then the identified protocol is http.

   As a best practice, if you import rules from files named after the protocol, then do not specify the destination port number in the rules. If you do, the Sensor restricts its search to those port numbers.

2. If the Manager is unable to identify the protocol by the first method, it identifies the protocol based on the destination port number in the rule. This assumes that the destination port number is a standard port number. For example, if you import a rule from a generically named rules file but the destination port is 80, then the identified protocol is http.

   The Manager uses this method when you import a rule from a generically named rules file (for example, myrules.rules or when you create the rule in the Custom Attack Editor.

   As a best practice, specify the destination port number (standard or otherwise) if you import the rules from a generically named file. This can improve Sensor performance because even if the protocol is not identified, the Sensor looks for the pattern only for the matching destination protocol.

See also

Create a snort custom attack — an example on page 145
How to use snort rules to detect IP communication between specific hosts

This section explains how to construct Snort rules that can detect TCP, UDP, or ICMP communication between specific hosts or networks based on IP address.

To detect TCP, UDP and ICMP communication between a set of IP addresses, you must create 3 separate rules.

TCP communication

To detect TCP communication between hosts specify `ip_proto:tcp;` and `flags:S` as the attack parameters.

**Example:**

```plaintext
```

Here 10.1.1.10 is an example source IP and [192.168.230.0/24, 192.168.231.0/24] are the example destination subnets.

If you do not specify `flag:S` the rule is triggered for all the TCP packets between the specified hosts.

UDP communication

To detect UDP communication between hosts, specify `ip_proto:udp;` as the attack parameter.

**Example:**

```plaintext
alert udp 10.1.1.10 any -> [192.168.230.0/24, 192.168.231.0/24] any (msg:"Malicious UDP Traffic";ip_proto:udp;sid:3122;)
```

Here 10.1.1.10 is an example source IP and [192.168.230.0/24, 192.168.231.0/24] are the example destination subnets. The Sensor sends alerts for any UDP packet detected between the specified hosts.

ICMP communication

To detect ICMP communication between hosts, specify `ip_proto:icmp;` as the attack parameter.

**Example:**

```plaintext
```

Here 10.1.1.10 is an example source IP and [192.168.230.0/24, 192.168.231.0/24] are the example destination subnets.

In the rule, use the following parameters along with `icmp` for a more specific detection:

- `itype` - Use this to check for a specific ICMP type value:
  
  **Syntax:** `itype:[<|>]<number>[<><number>];`

- `icode` - Use this to check for a specific ICMP code value.
Syntax: icode: [<|>]<number>[<><number>];

Example:

```
alert icmp 10.1.1.10 any -> [192.168.230.0/24, 192.168.231.0/24] any (msg:"example for itype
and icode"; itype 8; icode:0; sid:2022; priority:3;)
```

Here 10.1.1.10 is an example source IP and [192.168.230.0/24, 192.168.231.0/24] are the example destination subnets. This rule is triggered for inbound traffic where the type is 8 and code is 0.

### Importing snort custom attacks

Using the Custom Attack Editor, you can import Snort rules from a file against constructing them one by one. The import feature can be helpful when you want to create a large number of Snort Custom Attacks in one go. Also, sources such as the Snort user community, distribute files containing the rules, and you can import them all directly from the file instead of creating them one by one in the Custom Attack Editor.

You can use files to import Snort rules into the Manager following any one of the two methods described below:

- You can construct Snort rules in a file (extension, .rules) and then import that file into the Manager. The Manager validates the rules to see if they conform to the Snort rules syntax and then converts the valid rules into McAfee's proprietary format.

- A typical method, especially when importing rules from external sources, is to store the rules in different rules files based on a common criterion such as the protocol. Then import just a configuration file (extension, .conf), that in turn calls all the rules files. The conf file should contain the following:
  - References to the rules files that contain the rules that you want to import.
  - References to the classification files that contain the classification types used in the rules.

To reference a file from within another file, you use the include keyword. You can use the include keyword in .conf as well as .rules files. The syntax for using the include keyword is:

```
include <path to the file>/<file name>
```

Consider this scenario to understand how the include keyword is used.

If you are importing the rules distributed by the Snort user community, you would notice a sample configuration file named, snort.conf, which you need to modify to suit your requirements.

Consider a folder structure as shown below:
Assume that snort.conf is the configuration file you want to import, which is at D:\snort etc.

exploit.rules and ftp.rules are the files that contain the rules, and these are at D:\snort rules.

RULE_PATH is the variable that you want to use to denote the path to the two rules files.

For this scenario, snort.conf should have following entries at the appropriate places:

- Var RULE_PATH ../rules
- include $RULE_PATH/exploit.rules
- include $RULE_PATH/ftp.rules

If you do not want to use a variable for the path to the rules files, then the conf file should have the following entries:

- include ../rules/exploit.rules (or D:\snort\rules\exploit.rules)
- include ../rules/ftp.rules (or D:\snort\rules\ftp.rules)

Definitions of the variables used in the conf and rules files. If the file contains the same variables that are already available in the database, the values will be overwritten with the values from this file.

Tasks such as configuring DLLs, preprocessors, output plugins, and so on in the .conf file are not relevant in Network Security Platform.

**Snort rule validation utility**

When you import Snort Custom Attacks from a file, only the valid rules that contain the Snort features supported in Network Security Platform are converted successfully. You need to view the corresponding Conversion Notes in the Edit Snort Attack window for information on rules that failed to convert. Then, you must make the necessary changes to the rules and re-import them. After you fix the rules, make sure you use the Test Compile feature on them to verify their validity and compatibility.

You can validate the Snort rules through the Snort Rule Validation utility before you import them. This tool checks the rules for their validity and provides the details for those that will fail to import successfully. You can then correct the required rules and make sure that all your rules are successfully imported at the first attempt.

If you receive your Snort rules from a vendor, you can provide the Snort Rule Validation utility to your vendor. This will enable them to provide you validated rules, which you can subsequently import into the Manager. Another advantage is that you can use it offline to validate your Snort rules.

The location of the Snort Rule Validation utility is <Manager install directory> \App\diag\SnortUtil. Unzip the contents of McAfeeSnortStandAlone.ZIP on any Windows machine. The information on how to use this tool is in the README.txt, which is included in McAfeeSnortStandAlone.ZIP.

**Include duplicate Snort custom attacks**

When you create a Snort rule in the Custom Attack Editor or when you import Snort rules, the Custom Attack Editor checks if there is an equivalent McAfee attack signature based on the CVE IDs. If a McAfee attack signature exists, the corresponding Snort rule is referred to as a duplicate Snort custom attack or duplicate Snort rule. Such rules are imported into the Custom Attack Editor but excluded in the policies by default when you save them. You can manually change the State of such rules to Included.
Alternatively, you can specify your preference in the Custom Attack Editor such that, going forward the duplicate Snort custom attacks are included in the corresponding policies by default.

When you include duplicate Snort custom attacks and also enable alerting for the corresponding McAfee signature, then two alerts might be raised for the same attack traffic — one triggered by the Snort custom attack and the other by the McAfee attack.

**Task**

1. In the Custom Attack Editor, select File | Preferences.

![Figure 7-2 Include duplicate Snort custom attacks](image)

   **Figure 7-2 Include duplicate Snort custom attacks**

2. Select Include Duplicate Snort Attack.

   - This selection applies only to the Snort custom attacks that you will create or import from now on and not to the already excluded duplicate Snort custom attacks. You can only manually include the previously excluded duplicate Snort custom attacks by changing the State from the right-click menu of the corresponding attack.

   - This feature compares only the CVE IDs in Snort rules with the CVE IDs in the McAfee attack signatures.

   - The CVE ID that you mention in the Snort rule must be of the universal format for this feature to work.

<table>
<thead>
<tr>
<th>Correct</th>
<th>Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>reference:cve, 2010-0249;</code></td>
<td><code>reference:cve, CVE-2010-0249; The rule is included even though there is an equivalent McAfee signature because the Custom Attack Editor looked for CVE-2010-0249 instead of just 2010-0249.</code></td>
</tr>
</tbody>
</table>

**Import snort rules through a conf file**

**Before you begin**

Make sure of the following before you begin importing the conf file:

- You have all the rules files containing the rules that you want to import.

  The conf file has references to each rules file that you want to import. That is, the rules files are called from the config file using the include keyword and the absolute or relative path to the files. You can also use a variable to denote the path.

- All the variables, classifications, and references used in the rules are either defined in the conf or the rules files up front or available already in the Manager.
- Depending on how the rules have been constructed, the conf file may be referencing some files in addition to the rules files. Make sure all the files called by the conf file are in place.
- Each rule must have a globally unique Snort rule ID (SID) for it to be converted and saved in the Manager database. A rule is not considered for import, if the SID and revision number are same as that of a rule imported earlier. If the SID is same but not the revision number, then the rule with the highest revision number is retained.

This section provides the steps for importing Snort rules into the Manager using a conf file.

**Task**

1. Select **Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks**, and then click **Custom Attack Editor**.

2. In the **Custom Attack Editor**, select **File | Import | Snort Rules**.

3. In the Open dialog box, select the Files of Type as either All Files or Configuration Files [*.conf].

4. Navigate to the .conf file to be imported and click Open.

   All the rules are imported into the Manager and the valid ones are converted to McAfee's format. There could be some rules that were successfully converted to McAfee's format, some converted with warnings, and some that failed to convert.

5. You may want to see Viewing the imported Snort rules to:

   - View the details of the imported Snort rules
   - Know which rules converted successfully, which converted with warnings, and which failed to convert.

**See also**

- Variables on page 111
- Viewing the imported Snort Custom Attacks on page 122
- Define the classification types on page 113
Import snort rules through a rules file

**Before you begin**
Make sure of the following before you begin importing a rules file:

- All the variables, classifications, and references used in the rules are either defined in the rules files up front or already available in the Manager.
- Just like a conf file, a rules file too can call other files. So, make sure the files called by a rules file are in place.

You can import rules directly from a rules file without a conf file. This section provides the steps for importing Snort rules into the Manager using a rules file.

**Task**

1. Select **Policy** | **<Root Admin Domain>** | **Intrusion Prevention** | **Advanced** | **Custom Attacks**, and then click **Custom Attack Editor**.

2. In the Custom Attack Editor, select **File** | **Import** | **Snort Rules**.

3. In the Open dialog box, select the **Files of Type** as either **All Files** or **Rules Files[*.rules]**.

4. Navigate to the .rules file to be imported and click **Open**.

All the rules are imported into the Manager and the valid ones are converted to McAfee's format. There could be some rules that were successfully converted to McAfee's format, some converted with warnings, and some that failed to convert.

5. You may want to see Viewing the imported Snort rules to:
   - View the details of the imported Snort rules
   - Know which rules converted successfully, which converted with warnings, and which failed to convert.

**See also**

*Variables* on page 111
*Viewing the imported Snort Custom Attacks* on page 122
*Define the classification types* on page 113
Viewing the imported Snort Custom Attacks

All the newly imported rules are listed in a separate tab. This enables you to view which rules converted and which failed for you to take appropriate action. This section provides a brief description of the Custom Attack Editor for the purpose of understanding the properties of a Snort rule after import.

The column headings in the tabs of the Custom Attack Editor are customizable. What you see in the figure below are the default column headings. To customize a column heading, right-click on it and select Rename.

![Figure 7-3 Tab regions](image)

The following table describes the figure above:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Custom Attacks</td>
<td>This is the tab that lists all the custom attacks imported so far in addition to the custom attacks already saved in the database.</td>
</tr>
<tr>
<td>NSP Attack ID</td>
<td>Until you save the rules in the Manager database, the imported rules are assigned the same temporary id. When you save the imported rules in the database, the Manager assigns a unique attack id to each rule. For custom McAfee attacks, this ID starts with 0xc and for custom Snort attacks, it starts with 0xe. You can sort the listed attacks based on multiple columns by pressing the Ctrl key and clicking on a column heading. The number next to a column heading denotes the order in which the rules are sorted.</td>
</tr>
<tr>
<td>SID</td>
<td>Snort rule ID (SID) is the ID assigned to a Snort rule by you or the party that provided the rule. The Snort attack definitions that you want to save in the McAfee database must have a unique SID. Make sure that the SIDs of the attacks that you are writing or importing have not been used by the definitions that are already in the database. For attack definitions that failed to import, the McAfee assigns -1 as the SID.</td>
</tr>
<tr>
<td>Imported Snort Rule</td>
<td>This is the tab that lists the newly imported rules. If you import the rules from another file, then one more tab named, Imported Snort Rule is created which will list the rules from that file.</td>
</tr>
<tr>
<td>Last Modified</td>
<td>This is the time stamp when a rule was imported or modified.</td>
</tr>
</tbody>
</table>
Table 7-1  Tab regions *(continued)*

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
</table>
| State       | This column indicates whether a rule will be included in the Network Security Platform policies when you save the rules to the database. You can right-click on a rule and include or exclude it.  
At the time of importing a Snort rule, the McAfee checks if there is an equivalent McAfee attack signature based on the CVE IDs. If a McAfee attack signature exists, the Snort rule is imported but excluded in the policies by default. You can manually include such rules. Alternatively, in the Custom Attack Editor select File | Preferences and select Include Duplicate Snort Attack. Now, when you save the custom attacks, the Snort custom attacks, for which there is an equivalent McAfee signature, are also included in the corresponding policies.  
When you include such Snort custom attacks as defined above, and also enable alerting for the corresponding McAfee signature, then two alerts may be raised for this attack - one triggered by the Snort attack and the other by the McAfee attack. |
| Name        | The name for the attack that the Manager assigns post-import. The format that the Manager uses for the name is SNORT:<the text specified for the msg rule option in the rule>(<SID>). So this name is modified accordingly if you modify the msg text or the SID of the rule.  
Note that msg rule option and a unique SID are required in a rule for you to import it into the Manager. |
| Severity    | The Manager assigns a severity of low, medium, or high based on the priority value of the rule. For a rule, this priority could be derived on its classtype or the priority tag.  
The Manager assigns the severity for an imported Snort attack definition based on the following:  
• A priority 1 Snort attack definition is assigned a severity of High.  
• A priority 2 Snort attack definition is assigned a severity of Medium.  
• A priority of 3 or higher is assigned a severity of Low. |
| Format      | This column indicates whether it is a Snort rule or McAfee custom rule. |
| Conversion Result | This column indicates whether the rule was converted to McAfee. Some rules may have been converted but with warnings. For example, rules for which an equivalent McAfee signature exists are converted with warnings.  
Some rules may have failed to convert. For example, rules that use undefined variables fail to convert. |
| Conversion Notes | This column displays any error or warning messages related to the conversion. It may also displays the metadata of a rule, if available. |
| UDP Blocking | This column indicates that the Sensor would drop just the packet that matches the custom attack. In case of Snort Custom Attacks, it is always set to attack-packet. |
| Target Device | Specify the Sensor series to which the attack should be applied to. For example, if you specify I-series only, then only the I-series Sensors check for the corresponding rule.  
Snort rules that meet the following conditions are not supported on I-series Sensors. So, when you import such rules, the Target Device Type is automatically set to M-series and NS-series.  
• The Snort rule has the PCRE option but no content keyword in it.  
• The content keyword is not preceding the PCRE option in a rule.  
• The content keyword is preceding the PCRE option, but they are referring to different modifiers.  

Using the Display Filter

To locate custom attacks based on values such as severity and conversion result, you can just sort the listed attack definitions based on these values by clicking on the corresponding column heading. For advanced search capabilities, you can use the Display Filter feature. You can create a display filter with a set of criteria to view only those attack definitions that meet the criteria. For example, you can locate attack definitions based on what is contained in the attack names. This way you can easily locate specific attack definitions from a larger set.

Display filters are applicable only to the All Custom Attacks tab that lists all the custom attacks imported so far in addition to the custom attacks already saved in the database.

See also
Import snort rules through a conf file on page 119
Import snort rules through a rules file on page 121

Create a display filter

To understand how to create a display filter, consider that you want to view only those attack definitions that meet the following criteria:

- Name contains the following strings:
  - Byte Test
  - Greater Than
- State is Included
- Severity is High

To create a display filter for the above criteria:

**Task**

1. In the All Custom Attacks tab of the Custom Attack Editor, click Display Filter and then click New.
2. In the Display Filter window, enter a Display Filter name for your reference.
3. Under Filter Criteria, click on next to Attack Name.
4. On the right-hand side pane, select Contains from the drop-down menu and enter "Byte Test" and then click on the adjacent . A second row of options is displayed.
5. In the second row, click on the toggle button next to the drop-down menu until it turns to "and."
6. Select Contains from the drop-down menu and enter "Greater Than".
7. In the Filter Criteria pane, click on that is adjacent to State.
8. In the right-hand side pane, under State, select Equals and Included from the drop-down menus.
9. In the Filter Criteria pane, click on that is adjacent to Severity.
On the right-hand side pane, under Severity, select Equals and HIGH from the drop-down menus. Now, the Display Filter window should have the criteria as shown in the figure below.

![Figure 7-4 Display Filter](image)

Click Apply Once.

The attack definitions that meet the criteria set are displayed in a new tab.

**Correct the failed conversions**

Using an example, this section explains how to fix the rules that failed to convert.

To troubleshoot a failed rule:

**Task**

1. Double click on the rule in the Custom Attack Editor to open the Edit Snort Attack window.

2. Read the translation error.

   In this example, translation error is related to the reference key word. Reviewing the reference URL shows that it is specified as `www.zdnet.com.au/downloads/0,39024478,39111669s,00.htm`. The URL is corrected to `www.zdnet.com.au/downloads/0/39024478/39111669s/00.htm`. 
3 Click Validate to see if the error is corrected.

4 If the error is corrected, click OK and then click Save to save the changes you made. The rule is now converted to McAfee's format and stored in the Manager database.

After you correct the rules, use the Test Compile feature to check if they are valid and compatible. You can select the required attacks and select Test Compile from the right-click menu.

5 Select the corrected Snort custom attack and then select Attack | Test Compile the Attacks.

If the Snort custom attack is valid and compatible, then a success message is displayed. If not check if the rule contains any unsupported rule options and make the required changes. If you include a rule that failed to compile, the subsequent signature set update to the Sensors fail.

Troubleshoot rules converted with warnings

Some valid rules may convert but with Conversion Result as "warning". There are two main reasons why rules convert with a warning.

• Rules for which there is an equivalent signature in the Manager are converted with warnings. The Manager identifies such duplicates based on the CVE id mentioned in the rule and the signature. The State of such duplicate rules is "excluded". This means that these rules are not included in the rule sets. You can right-click on an excluded attack and change its State to included. You can specify your preference in the Custom Attack Editor such that, going forward the duplicate Snort custom attacks are included in the corresponding policies by default. For information on how to set this preference, see the Include duplicate Snort custom attacks section.

• Rules that contain or miss certain rule options are converted with a warning. Such rules are "included." However, some of them may impact Sensor performance. So, it is recommended that you investigate such rules to see if they can really impact Sensor performance. If so, then fix them before you include them in the rule sets. Select all such custom attacks in the Custom Attack Editor and then select Attack | Test Compile the Attacks to verify if the custom attacks are valid and compatible with Network Security Platform. If you include custom attacks that failed to test compile, the subsequent signature set update to the Sensors might fail.

Write snort custom attacks

Before you begin

Before you begin to construct the Snort rules, review the following:

• In the rule, you can use only the variables, classtypes, and references that are available in the Manager. For information on how to view the available values, see Viewing the variables and classification types.

• You can write only one rule at a time.
You can construct Snort rules directly in the Manager using the Custom Attack Editor. Note that these rules must conform to the Snort rules language syntax. Structure of a Snort rule provides information how to construct Snort rules within Network Security Platform.

To construct Snort rules in the Manager:

**Task**

1. Select **Policy** | **<Root Admin Domain>** | **Intrusion Prevention** | **Advanced** | **Custom Attacks** | **Custom Attack Editor**.

   The Custom Attack Editor opens with the existing Custom Attacks listed in the All Custom Attacks tab.

2. Select **Attack** | **New** | **Snort Rules**.

3. Construct the Snort rule in the Add Snort Rule dialog.

   Note the following:
   - You cannot define variables in the Add Snort Rule dialog, but you can use the variables that are available in the database.
   - You can write only one rule at a time.
   - If you are using the keywords such as classtype or reference, make sure the corresponding values are already defined and available in the database. For example, to import classifications, you can import a file that calls the classification config file and then use these classifications in the rules that you construct.

4. Click **Save**.

   The rule is listed in the All Custom Attacks tab.

5. Save the rule to the database so that it gets included in the relevant policies.

**See also**

*Saving the Snort custom attacks on page 127*

---

**Saving the Snort custom attacks**

After you write or import the rules, you need to save them to the Manager database. Then, the rules, for which the State is included, are automatically pulled into the various exploit policies (both for inbound and outbound). This is similar to how the McAfee custom rules (formerly known as User-Defined Signatures) are included in the policies. Review this section to understand how the rules are included in the policies.

- Only the rules that meet both these conditions qualify to be included in the exploit policies of Network Security Platform
  - The Snort rule has been converted to McAfee format successfully or with warnings.
  - The State is set to included.

- All the qualified rules are included in the All Inclusive policies. In addition, these rules are also included in other policies that meet the following criteria:
  - Benign Trigger Probability of medium. By default all the imported Snort rules are assigned a Benign Trigger Probability value of medium which cannot be modified.
  - Severity of the attack. This depends on the default classification based on the classtype of a rule or the priority tag in the rule.
  - Protocol.
Save the qualified rules

Before you save the Snort Custom Attacks to the database, you can also specify the Sensor type. For example, if you choose I-series as the Sensor type for an attack, then this attack definition is relevant only to the I-series Sensors of the corresponding admin domain. So only the I-series Sensors inspect the traffic for this particular attack definition. Even when you apply the same policy to an M-series Sensor and NS-series Sensor, it does not check for this attack. By default, when you import or create an attack in the Manager client, it is selected for I-series, M-series and NS-series.

Task
1. Go to the corresponding tab in the Custom Attack Editor.
2. Verify that the rules that you want to be included in the rulesets are in the "included" state. If not, you can right-click on the attack and select include.
3. Verify the Target Device to which the Snort Custom Attack should be applied to.
   For example, if you select I-series, then only the I-series Sensors in the current admin domain inspect traffic for this attack.
4. Click File | Save and then click OK to confirm.

The qualified rules are saved in the database and the corresponding policies are updated with these attacks. You can view the progress in the bottom part of the Custom Attack Editor.

5. You need to update the Sensors for the saved attacks to be detected.
   You can also verify if the attacks are included in the policies.
   2. Double-click one of the All Inclusive policies; for example, you can open the All Inclusive With Audit policy.
   3. In the Exploit&Policy Violation Attacks tab, double-click on All Protocols.
   4. Sort the attacks based on Name and verify if the Snort rules have been included.

Customizing the snort rules attack responses

You can use the Policy Editor to customize the Snort attacks like you would customize any other McAfee attack. If you make any configuration changes to an attack, you should update the Sensor of the changes for them to take effect.

By default, when you save a Snort attack to the database, Enable Attack and Enable Alert are enabled. All other configurations including blocking and packet logging are disabled.
Delete the snort rules from the database

Whenever required, you can delete the snort rules from the database and stop the Sensors from sending alerts:

**Task**

1. Open the Custom Rules Editor.

2. All the custom rules in the database are listed.

3. Select the required rules, right-click and select **Delete**.

4. Click **File | Save** to save the changes and update the policies.

5. Update the Sensors of the configuration changes.
Management of snort custom attacks
Delete the snort rules from the database
Common tasks

This section contains information on tasks that are common to both McAfee as well as Snort Custom Attacks.

Contents

- How attacks are included in policies
- Viewing a policy to verify inclusion of the attack
- Verify the inclusion of custom attack in a rule set
- Add attack descriptions to the Attack Encyclopedia
- Compile the attack definitions
- Update the Sensor configuration to apply a policy
- Custom attacks export
- Export the custom attacks

How attacks are included in policies

After you create a Custom Attack and the constituent signatures or rules, you need to save the attack in the Manager server for it to be included in every rule set where the attack definition fits one or more categories-rule set categorization was configured as part of attack creation. When you apply a policy that includes a rule set containing your attack definition, detection of your attack is active.

Consider that when you created a McAfee Custom Attack, you set the Severity to Medium and chose HTTP as the Exploit classification. Once exported, this attack is included in all rule sets that include Medium-severity, HTTP attacks, such as the Default, Outside Firewall, and Web Server rule sets provided with McAfee® Network Security Platform. (This also includes any rule set that you have created which calls for HTTP attacks of Medium severity or higher.) When you apply a policy that includes one of these rule sets, you are applying your Custom Attack for active searching and alerting upon detection.

When you save the custom attacks to the Manager server, only those attacks with State as "Included" are included in the rule sets. If you want to change the State of a custom attack, right click on it and select Include or Exclude.

When you create a McAfee Custom Attack, the State is set to "included" by default. In case of Snort Custom Attacks, there are instances where the state is set to "excluded." For example, if there is a McAfee attack signature with the same CVE ID, the Manager sets the State to excluded when you save the attack. The State is also set to excluded if the Conversion Result is failed or warning.

If after saving the Custom Attack to the Manager server, you import the attack back to the Custom Attack Editor, edit the file and make a name change to either the attack or a signature in the attack, then save it again in the Manager server, the name change will not take affect in any open Threat Analyzer views. You must close all Threat Analyzer windows and restart the Threat Analyzer to see the name change upon attack detection.
Viewing a policy to verify inclusion of the attack

After you save the custom attacks in the Manager server, verify if the attacks in "included" State have actually been included in the policies. The attacks are categorized based on the following and then included in the corresponding rule sets.

When you have finished creating a custom attack, you need to save it. Saving the attack in the Manager server, includes your attack definition in every rule set where your attack fits one or more categories, provided the State of the attack is "included." When you apply a policy that includes a rule set containing your attack, detection of your attack is active.

The Manager categorizes attacks based on:

- Impact application
- Impact operating system
- Severity of the attack
- Benign Trigger Probability (BTP)
- Impact application layer protocol

For example, when you created a McAfee Custom Attack, you set the Severity to Medium and chose HTTP as the Exploit classification. Once saved in the Manager server, the attack is included in all rule sets that include Medium-severity, HTTP attacks, such as the Default, Outside Firewall, and Web Server rule sets provided with McAfee® Network Security Platform. (This also includes any rule set that you have created which calls for HTTP attacks of Medium severity or higher.) When you apply a policy that includes one of these rule sets to a port or VIDS, you are including your attack for IPS.

If you make a name change to either the attack or a signature within the attack, then save it back into the Manager, the name change will not take affect in any open Threat Analyzer views. You must close all Threat Analyzer windows and restart the Threat Analyzer to see the name change upon attack detection.

See also

- Create custom attacks on page 41
- Create a signature on page 44
- Import previously exported custom attacks on page 64
- Update the Sensor configuration to apply a policy on page 137
- Create a snort custom attack — an example on page 145
- Add the condition on page 143
- Add the second condition with OR comparisons on page 151
Verify the inclusion of custom attack in a rule set

To verify if a custom attack has been included in a rule set:

**Task**


2. Select the "All Inclusive with Audit" rule set and click View Attacks Selected.
3 Select the protocol row from the table that matches the category of your custom attack and click View/Edit.

4 Scroll down the list (sorted alphabetically) to find your attack file. McAfee Custom Attacks have "UDS-" appended to the beginning of the name, while Snort Custom Attacks have "Snort-" appended to the beginning.

Add attack descriptions to the Attack Encyclopedia

To enter Attack Description and comments in the Attack Encyclopedia for custom attacks, use the Policy Editor/Rule Set Editor user interfaces. When an alert is raised for a custom attack, the same details can be viewed from the Threat Analyzer as well.

This procedure assumes that you have a McAfee or a Snort Custom Attack included in one or more rule sets.
Task

1. Open the custom attack in the IPS Policy Editor:
   a. When the McAfee Custom Attack is exported to Manager, select Policy | <Root Admin Domain> | Intrusion Prevention | IPS Policies.
   b. Select any policy name (for example, All-Inclusive With Audit), and then click View/Edit.
   c. In the Edit IPS Policy window, select Exploit Attacks tab, and then select the application layer protocol that the attack impacts.
      • In case of McAfee Custom Attacks, this is the protocol that is specified in the Matching Criteria section of the Custom Attack Editor.
      • In case of Snort Custom Attacks, the Manager identifies the protocol for a rule.
      • If you are not sure of the protocol, select All Protocols.

   To view all the UDS attacks, AIWA policy needs to be applied on Interface.

2. In the Configure Attack Detail window, locate and select the custom attack. To locate the attack, you can check the name under Attack Name column.

3. Select the custom attack and then double-click it.

   The Edit Attack Detail window is displayed.
4 Click **Attack Desc.** to view the details of the custom attack from the attack encyclopedia.

5 Click **Annotate Desc.** to enter the attack description in the attack encyclopedia.
Compile the attack definitions

In the policies, you can use the attack definitions from the following types:

- McAfee-supplied Attack definitions (signature set)
- McAfee Custom Attack
- Snort Custom Attack

There can be instances where you may want to include only specific types of attack definitions for a specific Sensor. For example, you may want to configure a Sensor to monitor traffic for specific attacks. You can also use this feature to troubleshoot and isolate the attack definitions that may cause an adverse effect on a Sensor's performance.

To specify the attack definition type for a Sensor:

Task
1 Select Devices | <Device Name> | Troubleshooting | Attack Compilation.
2 Select the attack definition type:
   - **Default McAfee Attacks** (from signature set): Select this if you want to use the definitions from the signature set in the Manager.
   - **Custom Attacks - McAfee Format**: Select this if you want to use the definitions that you created in the McAfee format. This also includes the McAfee-supplied custom attacks (emergency UDS).
   - **Custom Attacks - Imported Snort Rules**: Select this if you want to use the Snort Custom Attack definitions that you created or imported.
3 Click Save.

When you update a Sensor with the configuration changes, only the attack definitions from the type specified here are pushed to the Sensor.

Update the Sensor configuration to apply a policy

Once you have verified that a custom attack is included in a rule set, you need to update the Sensors of the changes for the Sensors to look for traffic matching the attack definition. Since addition of a custom attack affects one or more rule sets, which in turn affects one or more policies, your policies may require updating across all of your McAfee® Network Security Sensors (Sensor)s in order for effective detection of your attack definition.

The **Device Configuration Changes** action sends configuration changes, signature updates, and policy changes to all of the devices under the **Devices** node.

The **Device Configuration Changes** action updates multiple Sensors, but only transmits the update to one device at a time.
To update the configurations of multiple devices:

**Task**

1. Select **Devices | <Admin Domain Node> | Global | Deploy Configuration Changes**.

   You can also do it at the device level by selecting **Devices | <Admin Domain Node> | Device | <Device Name> | Deploy Configuration Changes**.

2. Select the Sensor(s) to be updated. The fields are described as follows:
   - **Device Name**: Unique name of each device.
   - **Last Update**: Last day and time device configuration was updated.
   - **Updating Mode**: Online or offline update mechanism selected for the device.
   - **Pending Changes**: Summary of changes that have been made.
   - **Configuration & Signature Set**: A selected checkbox indicates that the device is to be updated for any configuration change other than those related to SSL key management.
   - **Status**: Displays the status of the configuration update.

![Figure 8-1 Device Configuration Changes page](image)

3. Click **Update**.

   A Snort Custom Attack definition being applied to a Sensor port depends on the following:
   - The attack definition should be included in the corresponding policy.
   - The attack definition type that you have specified.
   - You should have specified the corresponding Target Device for the attack definition. That is, if it is an I-series Sensor port, you should have selected I-series or M-series and NS-series, and I-series as the Target Device for the attack definition.

**See also**

*Viewing a policy to verify inclusion of the attack on page 132*

---

**Custom attacks export**

You can export all the McAfee Custom Attacks, Snort Custom Attacks, and custom-defined grep protocols in the Manager to a ZIP file. The export feature enables you to use the custom attacks on a different Manager without having to re-create them. McAfee strongly recommends that you do not try to modify the exported attack files and then re-import them.

In the exported ZIP file:

- The McAfee Custom Attacks are stored in the attacks.xml file.
- The packet grep protocols are stored in the pktgrepprotocol.xml file.
The Snort Custom Attacks are exported in the Snort format.

The Snort Custom Attacks Rules and the related data such as variables that you imported are stored in correspondingly named files. For example, assume that you had imported rules from ftp.rules file. When you export, these rules are stored in ftp.rules file within the ZIP.

The rules that you directly created in the Editor are contained in unknown.rules file.

There is no option to export just the McAfee Custom Attacks or just the Snort Custom Attacks. When you export, all custom attacks listed in the All Custom Attacks tab and the custom-defined packet grep protocols are exported.

### Export the custom attacks

To export the custom attacks to your client or other reachable location, do the following:

**Task**

1. In the Custom Attack Editor, select File | Export | Custom McAfee Attacks.
2. Select the folder where you want to save the ZIP file.
3. Type a name for the zip file. Do not add an extension.
4. Click Save.
Common tasks
Export the custom attacks
Examples

This section provides information on how to create some example Custom Attack definitions. The focus of this section is on McAfee Custom Attack definitions. However, for each example, the equivalent Snort Custom Attack is also provided for your reference.

Contents

- Custom attack with a pattern-match signature
- Custom attacks with multiple comparisons

Custom attack with a pattern-match signature

**Scenario**

This is about creating a Custom Attack Definition that uses pattern matching to detect any HTTP GET requests in the URI path for a specific CGI script. This scenario is useful because it illustrates the simplest method of configuring a custom attack definition—with proper syntax—defined options, and case sensitivity.

Creation of such an attack definition is also a good example of activity that may not be a malicious attack; rather, you could create such an instance and use it to track requests for sensitive information.

In order to write a proper McAfee Custom Attack or a Snort Custom Attack for this example, you must identify several key elements:

- application protocol: HTTP
- where to look: URI path
- detection window: request
- request method: get
- string to match: cgi.bin/trillion.pl or cgi.bin/trilliant.pl, where only "pl" is case insensitive

When you have all of the required elements to properly identify the activity, attack definition can be successful.

To create a custom attack definition:

- Open the Custom Attack Editor.
  
  Select Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks | Custom Attack Editor.

**Create a custom attack**

This section explains how to create a McAfee Custom Attack definition for the scenario explained in the previous section.
Create an attack definition

To create the attack definition for this example:

**Task**

1. In the Custom Attack Editor, select **Attack | New | McAfee Attack Definition | Exploit Attack**.
   
   The Add Exploit Attack window opens.

2. Select one of the options and click **Next**.

3. You can specify a name such as "CGI: Trillion or Trilliant" as the **Attack Name**.
   
   The letters "UDS" (user-defined signature) are appended to the front of the name upon completion; thus, this attack appears as "UDS-CGI: Trillion or Trilliant" in the Custom Attack Editor, as well as the attack database when you save in the Manager server.

4. (Optional) Type a description for your attack. This area can be used for your notes or other specific information pertinent to your new attack.

5. You can select Medium as the **Severity** because this example scenario does not necessarily involve malicious activity.

6. Select Advanced Protection Options/Web Application Server as the **Protection Category**.

7. In the Matching Criteria tab, click **Add**.

8. In the Add Protocols Packages window, select **Protocol Only**, then click **Next**.

9. Because this scenario is HTTP-based, select **HTTP** from the Protocol List and click **Finish**.

   When you subsequently save this attack definition in the Manager server, it will be part of every rule set which includes Medium-severity, HTTP attacks (must match both severity and protocol).

10. Attack details configuration is complete. Continue to create signature.

Create the signature

After you create the attack definition, you create the signature for the attack.
**Task**

1. In the Add McAfeeExploit Attack window, click **Signatures** and then click **New**.

   The **New Signature** window is displayed.

   ![New Signature window](image)

   **Figure 9-1  New Signature window**

2. (Optional) Clear the **Signature Name** and type a new name for your signature.

3. For this example, you can leave the **Benign Trigger Probability** and **Target Host Architecture** with the default values.

4. This example is to search for a string in the HTTP URI. So, select **request** as the **Detection Window**.

5. Based on the Sensor model that you plan to use for this example, select the **Target Device Type**.

6. Add the condition to the signature.

   It is in the conditions that you specify the following details:
   - The string that the Sensor should look for.
   - Which section of the http request should it look for the string.

7. Proceed to Adding the Condition.

**Tasks**

- Add the condition on page 143

**See also**

*Signature creation interface on page 27*

**Add the condition**

To add the condition to the example signature:

**Task**

1. In the Add Exploit Attack window, click ADD in the Conditions section.

   Condition 1 appears in the white box.

2. Click Condition 1 so that it is highlighted.
3 Click AND under in the Comparisons section.
   The Select Comparison dialog opens.

   ![Figure 9-2 Adding Conditions](image)

4 For this example, select **String Pattern Match**, then click **Next** in the Select Comparison dialog.

5 Select **HTTP** from the Protocol List, and then click **Next**.
   Because you selected HTTP, the Custom Attack Editor displays the HTTP-specific protocol fields on the following screen.

6 Configure the fields for the comparison you have chosen.
   For this example, specify `req-uri-path` for the Protocol Field. This specifies that the Sensor should search in the URI of the request packet.

7 Select **get** as the http request method.

8 From the drop-down list in the **Regular Expression to Match** section, select the matching criteria as **Equals** which means that the comparison criteria must be equal to the regular expression entered.

9 Type the pattern to match using the **Regular Expression to Match**.
   For this example, the pattern to match is either "cgi.bin/trillion.pl" or "cgi.bin/trilliant.pl", where "pl" is case insensitive. To properly write this expression, use the following rules:
   - Add a backward slash (\) before every dot (.) to escape: cgi.bin = cgi\bin
   - Use alternatives where possible. For this example, trillion and trilliant can be written as: trilli(on|ant).
   - Use character classes to denote case insensitivity for "pl": [Pp][Ll]

   The final string should appear as: cgi\bin/trilli(on|ant)\.[Pp][Ll]

   ![Figure 9-3 Configure Comparison window](image)
10 Click **Validate** to verify that your expression is a valid string, and all required options are represented. Click **OK** to close the validation message window.

11 Click **Finish**.

Your comparison appears under **Condition 1**.

12 Click **Save** in the Add Exploit Attack window.

13 Verify that the attack definition is listed in the All Custom Attacks tab.

14 Select **File** | **Save** to save the McAfee Custom Attack in the Manager server database.

15 Make sure the McAfee Custom Attack is saved in the database and also included in the policies.

**See also**
*Viewing a policy to verify inclusion of the attack on page 132*

**Create a snort custom attack — an example**

To write a Snort Custom Attack definition for the scenario explained in the previous section:

**Task**

1. In the Custom Attack Editor, select **Attack** | **New** | **Snort Rules**.
   
   The Add Snort Rule text box opens.

2. Construct the Snort rule for the scenario, which is:
alert tcp any any -> 192.168.1.1 80 (msg:"CGI: Trillion or Trilliant";content:"cgi.bin\trilli";http_uri;pcre:"/cgi.bin/trilli(on|ant).[Pp][Ll]/";priority:2;sid:2051;rev:1;)

**Figure 9-4 Add Snort Rule window**

Some points to note regarding the rule:

- Specifying the destination IP address improves the performance of the Sensor.

- Because you are adding the Snort rule directly in the Custom Attack Editor, as a best practice, you need to specify the destination port number. Then the Sensor checks only the http request packet for this rule.

- You need to specify the msg, priority, SID, and the revision number.

3. Click **Save**.
4 From the All Custom Attacks tab, double-click on the Snort Custom Attack that you just saved. The Edit Snort Attack window opens.

![Edit Snort Attack window](image)

**Figure 9-5  Edit Snort Attack window**

5 Note that the Manager assigns the Attack Name based on the msg option and SID of the rule.

6 Set the severity to Medium. This is because the priority of the rule is 2.

7 Note that the Protocol is set to http. This is because the destination port number is 80.

8 Click on the **Snort Rule** under the Snort Custom Attack.

![Edit Signature](image)

9 Note that the Benign Trigger Probability and the Target Host Architecture are all set to the default values.

10 Note that the Detection Window is set to Request because the destination port number is 80.
11 Based on the Sensor model that you plan to use for this example, select the **Target Device Type**.

12 Close the Edit Snort Attack window.

13 In the All Custom Attacks tab, select **File | Save** to save the Snort Custom Attack in the Manager server database.

14 Make sure the Snort Custom Attack is saved in the database and also included in the policies.

**See also**

- Define the classification types on page 113
- Identification of the protocol of a snort custom attack on page 115
- Viewing a policy to verify inclusion of the attack on page 132

---

**Custom attacks with multiple comparisons**

In case of McAfee Custom Attacks, can use multiple conditions and/or comparisons within a single attack definition to increase your confidence in detecting an unknown attack. There are many configuration possibilities; the procedure that follows details the following scenario:

- One signature with two conditions. Both conditions must be met before an alert is generated.
- The first condition has a single comparison.
- The second condition employs an OR comparison; that is, either of the two comparisons in the condition signify a positive match.

You can use various comparison methods within a condition. For example, if you employ two comparisons within a condition, one comparison can be a string match, while the second can be a fixed field check.

To create the McAfee Custom Attack definition:

- Open the Custom Attack Editor.

Select **Policy | <Root Admin Domain> | Intrusion Prevention | Advanced | Custom Attacks**.

**Create an attack definition for the example**

To create the attack definition for this example:

**Task**

1 In the Custom Attack Editor, select **Attack | New | McAfee Attack Definition | Exploit Attack**.
   The Add Exploit Attack window opens.

2 Select one of the options and click **Next**.

3 You can specify a **Attack Name**.
   The letters "UDS" (user-defined signature) are appended to the front of the name upon completion when you save it in the Manager server.

4 (Optional) Type a description for your attack. This area can be used for your notes or other specific information pertinent to your new attack.

5 Select High as the **Severity** for this example.

6 Select Advanced Protection Options/Web Application Server as the **Protection Category**.
In the Matching Criteria tab, click **Add**.

In the Add Protocols Packages window, select **Protocol Only**, then click **Next**.

Select **HTTP** from the Protocol List and click **Finish**.

Click **Add** again in the Matching Criteria tab.

Select **Software Package** and then click **Next**.

Select **IIS** from the Software Package List and then click **Finish**.

Verify that both HTTP and IIS appear in the Matching Criteria table.

Attack details configuration is complete. Continue to create signature.

**Tasks**
- Create the signature for the example on page 149

**Create the signature for the example**
After you create the attack definition, you create the signature for the attack.

**Task**
1. In the Add Exploit Attack window, click on **Signatures** and then select **New**.

![New Signature](image)

2. (Optional) Clear the **Signature Name** and type a new name for your signature.

3. For this example, you can leave the **Benign Trigger Probability** and **Target Host Architecture** with the default values.

4. For this example select **request** as the **Detection Window**.

5. Based on the Sensor model that you plan to use for this example, select the **Target Device Type**.

6. Proceed to Adding the first condition and comparison for your signature.

**Tasks**
- Add the first condition and comparison for your signature on page 150
- Add the second condition with OR comparisons on page 151

**See also**
- Signature creation interface on page 27
Add the first condition and comparison for your signature

To add the condition to the example signature:

**Task**

1. In the **Add Exploit Attack** window, click **ADD** in the Conditions section.
   - *Condition 1* appears in the white box.

2. Click **Condition 1** so that it is highlighted.

3. Click **AND** under in the Comparisons section.
   - The Select Comparison dialog opens.

4. For this example, select **String Pattern Match**, then click **Next** in the Select Comparison dialog.

5. Select **HTTP** from the Protocol List, and then click **Next**.
   - Because you selected HTTP, the Custom Attack Editor displays the HTTP-specific protocol fields on the following screen.

6. Configure the fields for the comparison you have chosen.
   - For this example, specify **req-uri-path** for the Protocol Field. This specifies that the Sensor should search in the URI of the request packet.

7. Select **get** as the http request method.

8. From the drop-down list in the **Regular Expression to Match** section, select the matching criteria as **Equals** which means that the comparison criteria has to be equal to the string pattern entered.

9. In the **Regular Expression to Match** text box, type the pattern to match using the Regular Expression Language rules.
   - The pattern to match is "private.exe". You must add a backward slash (\) before the dot (.) to escape the dot character so that it is properly interpreted. The final string should appear as: `private\exe`.

**Figure 9-7 Adding Conditions**

**Figure 9-8 Adding the string pattern to match**
10 Click **Validate** to verify that your expression is valid, and that the appropriate pattern is represented.

11 Click **Finish** when done with the "Configure Comparison" fields.

Your comparison appears under **Condition1**.

![Figure 9-9 Condition with the string pattern comparison](image)

**Add the second condition with OR comparisons**

To add the second condition to the example signature:

**Task**

1. In the **Add Exploit Attack window**, click **ADD** in the Conditions section.

   The first condition is minimized and the second condition reads as **[AND THEN] Condition 2**, thus signifying the first condition must match before the second condition can be tested.

   ![Figure 9-10 Adding conditions](image)

2. Select the text of the second condition, **OR** from **Comparisons**.

3. Select **Calculated Numerical Value Comparison** from the Comparison List, and then click **Next**.

4. Select **HTTP** from the Protocol List, and then click **Next**.

5. Select **req-uri-length** for the Protocol Field, then **get** as the request method.

6. Select **>** from the center drop-down list.

   ![Figure 9-11 Configure Comparison window to select the comparator](image)

7. (Optional) Check the **Signed** check box if you want the value to be signed. The default is unsigned (empty box).
8  Type a length value in the Value field. For this example, type 200 (bytes) as the length over which this comparison will match.

9  Click Finish; you are returned to the New Signature window.

10 Verify that your newly added comparison appears under Condition 2 under the heading [AND] (One Of).

![Figure 9-12 View the new condition defined](image)

11 Click on [AND] (One Of), then click OR under Comparisons.

12 Select Calculated Numerical Value Comparison from the Comparison List, and then click Next.

13 Select HTTP from the Protocol List, then click Next.

14 Select req-header-length for the Protocol Field, keep the default (ANY) as the http-req-hdr-type, then select get as the http-req-method.

15 Select > from the center drop-down list.

16 Type a length value in the Value field. For this example, type 1 (byte) as the length over which this comparison will match.

   As stated in the Description field, a value of 1 byte is significant for this comparison as the normal header length of a request should be zero.

17 Click Finish; you are returned to the New Signature window.
18 Verify that your newly added comparison appears beneath the first comparison you configured for Condition 2. The second comparison line is preceded by \[\text{OR}\] to signify the either-or relationship between the two comparisons.

![Comparison Diagram](image)

19 Click OK in the New Signature window.

Though there are multiple conditions and comparisons, only one signature was created, thus only one signature is uploaded to the Manager.

20 Verify that the attack definition is listed in the All Custom Attacks tab.

21 Select File | Save to save the McAfee Custom Attack in the Manager server database.

22 Make sure the McAfee Custom Attack is saved in the database and also included in the policies.

**See also**

*Viewing a policy to verify inclusion of the attack on page 132*

**Equivalent snort custom attack**

It is not possible to create an equivalent Snort Custom Attack for all the conditions used in this example, because Snort does not support http-get-req-header-length option.
Examples
Custom attacks with multiple comparisons
Management of custom attacks from the Central Manager

Using the McAfee® Network Security Central Manager, you can manage custom attacks for the corresponding Managers. Functionally, the Custom Attack Editor of a McAfee Network Security Central Manager (Central Manager) is the same as that of a Manager. So, see the earlier sections in this document for information on how to use the Editor for McAfee Custom Attacks and Snort Custom Attacks.

The summary of managing custom attacks from the Central Manager is as follows:

1. Create the custom attack definitions using the Custom Attack Editor in the Central Manager.
2. Save the custom attacks in the Central Manager server database.
3. Synchronize the policies of the constituent Managers.
   a. From the Resource Tree of the Central Manager, select Manager List | Policy Synchronization.
   b. Select the Synchronization Type for the required Managers and then click Synchronize.
4. View the status of the synchronization. Check the Status of Activities section on the Home page of the Central Manager.
5. Update the Sensors with the configuration change.
   a. Log on to the required Manager.
   b. Update the relevant Sensors with the policy change.
   c. Repeat this process for the other Managers.

See also
Management of snort custom attacks on page 4

Important notes

This section lists some important notes related to managing custom attacks in the Central Manager.

- All custom attacks sent from the Central Manager to the Manager as part of synchronization are automatically saved in the Manager database and also included in the relevant policies.
- In the Custom Attack Editor of the Manager, you cannot modify or delete any custom attack that was sent from the Central Manager. You cannot change the State of the Central Manager attack definitions in the Manager.
- When you synchronize the Central Manager with the constituent Managers, only the custom attacks with State as included are sent to the Managers.
• If you remove a Manager from the Central Manager, all custom attacks sent from the Central Manager to the Manager are removed from the Manager database. To remove these attacks from the corresponding Sensors as well, you need to do a configuration update.

• Recall the custom attack capacity per Sensor type:
  • You can use up to 4500 McAfee Custom Attacks and 2500 Snort Custom Attacks per I-series Sensor.
  • You can use up to 4500 McAfee Custom Attacks and 5000 Snort Custom Attacks per M-series and NS-series Sensor.

Factor this in when you send custom attacks from the Central Manager to the corresponding Managers.
# Index

**A**
- about this guide 7
- attack creation interfaces 25
- attack definition; example creating 148
- attack definitions 16
  - compiling 137
  - creating 142
  - reasons 9
- attack description 134
- attack detection 15
- attacks included in policies 131

**C**
- common tasks 131
- conventions and icons used in this guide 7
- custom attack
  - creating 13, 14, 42, 141
  - creating signature 44
- custom attack creation interfaces 25–27
- custom attack definition
  - configuring; multiple attacks correlation 64
- custom attack definitions
  - testing 16
- custom attack editor 11, 17
  - audience 11
  - creating display filter 24
  - default page 18
  - interface 17
  - menu items 19
  - tab regions 21
  - viewing options 23
- custom attack mechanics 35
- custom attack with pattern-match signature 141
- custom attacks 9
  - considerations 40
  - creating 13, 41
  - managing 41
  - types 10
- custom attacks export 138, 139
- custom attacks from the Central Manager 155
- custom attacks templates 47
  - detecting DNS query or response 53
  - detecting email attachment by file name 50
  - detecting string in application; custom port 55
  - detecting TCP connection attempts 58
  - detecting URL 47
- custom attacks with multiple comparisons 148
- custom reconnaissance attack definition
  - configuring 61
- custom reconnaissance attack; limitations 73

**D**
- display filter 124
- documentation
  - audience for this guide 7
  - product-specific, finding 8
  - typographical conventions and icons 7

**E**
- equivalent snort custom attack 153
- examples 141
- exported custom attacks; importing 64

**F**
- failed conversions; correcting 125
- first condition and comparison for signature; adding 150

**I**
- impact packages and protocols 14
- Important notes 155
- inclusion of custom attack verification 133
- intrusion prevention 14

**M**
- McAfee ServicePortal, accessing 8
- McAfee-defined custom attacks; importing 65

**P**
- packet search protocol interface 28

**Q**
- qualified rules; save 128
Index

R
regular expression language 70
regular expression syntax 71–73
rule header section 76
rule options 78
  general 78
  non-payload 101
  payload 80
  post-detection 106
rules converted with warnings 126

S
second condition with OR comparisons; adding 151
Sensor configuration download; policy application 137
ServicePortal, finding product documentation 8
signature
  create an example 142
  creating; adding condition 143
signature structure 35
signature terminology 10
signature test reference 37
  performance issues 40
signature; example
  creating 149
snort custom attack interfaces 31
snort custom attack mechanics 75
snort custom attacks 107
  best practices 110

snort custom attacks 107 (continued)
  considerations 107
  creating 111
  defining classification types 113
  defining variables 111, 112
  identifying protocol 115
  importing 117
  saving 127
  viewing variables and viewing classification types 113
  writing 126

snort custom attacks import
  viewing 122
snort rule structure 76
snort rule validation utility 118
snort rules attack responses; customization 128
snort rules deletion 129
snort rules import 119, 121
snort rules; IP communication 116

T
technical information 16
  technical support, finding product information 8

V
verification of attack inclusion; viewing policy 132