Using Expert Rules

What are Expert Rules

Expert Rules are text-based custom rules that can protect specified resources from unauthorized access and prevent exploits from known attacks. The system administrators can configure the Expert Rules in the **Exploit Prevention** policy available within **Threat Prevention** and enforce it to the endpoints.

Expert Rules provide additional parameters and allow much more flexibility than the custom rules you create in the **Access Protection** policy.

The **Trellix** predefined Expert Rules available in **Exploit Prevention** policy can be:

- enabled or disabled
- customized to Block and Report or Report only

You can write your own Expert Rules by understanding these **Trellix** proprietary syntaxes along with basic knowledge on the **Tool Command Language** (Tcl) programming:

Arbitrary Access Control (AAC)	**Legacy McAfee Host IPS-based Expert Rules**
Arbitrary Access Control (AAC) is a **Trellix** proprietary technology in **Threat Prevention** protect key resources. You can extend this protection by creating rules to protect specific files, processes, and registry items. AAC-based Expert Rules use a new syntax from the **Tool Command Language** (Tcl) interpreter version 7.6. Expert Rules enforced for: • **Files** — Protects files from unauthorized access. • **Processes** — Prevent the specific programs and processes from tampering and terminating . • **Registry** — Protects registry keys and registry values from unauthorized access. You can also create custom Files, Processes, and Registry rules in the **Access Protection** policy in **Threat Prevention**. But, these rules don't provide the complete functionality available with Expert Rules.	These Expert Rules follow the same syntax as rules created using the Expert method in **McAfee Host IPS**. **Trellix ENS** supports these **legacy class types**: • **Buffer Overflow** — Prevents buffer overflow exploits for applications in the Application Protection list. • **Illegal API Use** — Prevents illegal use of the Exploit Prevention API. The Expert Rules can only extend the functionality of the Illegal API Use signatures provided by Exploit Prevention content. This rules can't see APIs that aren't already covered in an Illegal API Use signature available in content. • **Services** — Protects Windows Services (Windows versions 8.0 and earlier only). You can also create custom Services rules in the **Access Protection** policy in **Threat Prevention**. But, these rules don't provide the complete functionality available with Expert Rules.

For more information about the Expert Rules commands in detail, see Learn Expert Rules commands for Files, Processes, and Registry.	For more information about the Expert Rules commands in detail, see Learn Expert Rules to protect Buffer overflow, Illegal API use, and Services.

Note

Each Expert Rule supports only one rule engine type. You can't mix different rule engine types in the same rule. That is, you can't combine a **McAfee Host IPS**-based rule (for example, Illegal API Use), with AAC-based rule (for example, Files).

Expert Rule types and supported syntaxes

Trellix ENS provides two syntaxes for creating the different Expert Rule types.

Rule type	AAC-based syntax	**Legacy** McAfee Host IPS**-based syntax**
Files	✓	✗
Registry	✓	✗
Processes	✓	✗
Buffer Overflow	✗	✓
Illegal API Use	✗	✓
Services	✗	✓
Program (**McAfee Host IPS** only)	✗	✗

The new AAC **Processes** rule type replaces the **McAfee Host IPS Program** rule type, which is not supported in **Trellix ENS**.

Note

You can't create Network IPS Expert Rules.

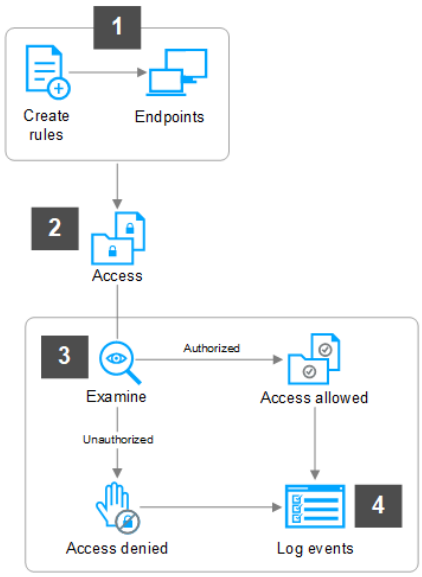
How Expert Rules work

Threat Prevention enforces Expert Rules on the client system the same as any other rule.

The signatures in the Exploit Prevention content provide default protection from **Trellix Labs**. If you need to protect additional resources, you can create custom rules in the Access Protection policy. For even further customization, create Expert Rules in the Exploit Prevention policy.

Here is the workflow of Expert rules:

1. An administrator creates the Expert rules and enforces them on the client system or self-managed endpoints.
2. A user or application tries to access the specific object on which the Expert rules are enforced.
3. Rules examine and perform one of these actions: a) allow access if user identity, access types, and other match values comply with the rule. b) block access if user identity, access types, and other match values do not comply with the rule.
4. Log events in the Event Log page of ENS.



Expert Rules to protect files

Create Expert Rules to protect Files using ePO

Expert rules control the access to files in a specific file path. Based on the access permission you set in the rule, it blocks and triggers an event, if any unauthorized source access the protected file.

Task

1. **Select Menu → Policy → Policy Catalog, then select Endpoint Security Threat Prevention from the Products list in the left pane.**

2. **From the Category list in the right pane, select Exploit Prevention.**
3. **Click the Edit link for an editable policy.**
4. **Click Show Advanced.**
5. **In the Signatures section, click Add Expert Rule.**
6. **In the Expert Rules Properties page, complete the fields.**
   ENS assigns the ID number for the rule automatically starting with 20000.
   a. **In the Rule Name, provide a unique name for the Expert rule.**
   b. **Select the severity level according to the Expert rule.**
      The severity provides information only; it has no effect on the rule action.
   c. **Select Block and Report actions for the rule by selecting the corresponding checkboxes.**
      Trellix recommends selecting **Report** action for initial validation. You can select **Block** and **Report** check boxes after validating that the rule triggers the appropriate events.
   d. **Select the Use Expert Rule template checkbox. This populates a template rule in the Rule content box based on the Rule type you select.**
      To get a blank template for writing the Expert rules, deselect **Use Expert Rule template**.
   e. **Select Files in the Rule type drop-down list.**
7. **Save the rule, then save the settings.**
8. **Validate the new policy on a client system.**
9. **Enforce the policy on the client systems.**

Create Expert Rules to protect Files on a client system

You can create Expert rules directly on a client system or self-managed endpoints that aren't managed by **Trellix ePO - On-prem**.

Before you begin

Make sure that the interface mode for the **Trellix Endpoint Security (ENS) Client** is set to Full access or log on to the **Trellix Endpoint Security (ENS) Client** as administrator.

Task

1. **Launch the Trellix Endpoint Security (ENS) Client.**
2. **Click Threat Prevention on the main Status page.**
   Or, from the **Action** menu , select **Settings**, then click **Threat Prevention** on the **Settings** page.
3. **Click Show Advanced.**
4. **In the Signatures section:**
   - Create a rule — Click **Add Expert Rule**.
   - Edit an existing user-defined rule — Double-click the rule in the table.
5. **In the Expert Rule Checker page, complete the fields.**
   Trellix ENS assigns the ID number automatically starting with 20000.
   a. **Select the severity and action for the rule.**
      The severity provides information only; it has no effect on the rule action.
   b. **Select Files in the Rule Type drop-down list.**
   c. **Add the rule code to the Rule content field.**
6. **Save the rule, then save the settings.**

7. **Validate the new Expert Rule on the client system.**

Expert Rule syntax to protect Files

To write an expert rule to protect your files, you need to ensure that it follows the correct syntax. Expert rules consist of process, one or more targets, and the matching conditions that rules must examine before allowing the source to access the files.

Here is a sample Expert rule for registry rule type and their respective definitions:

⚠ Caution

Expert Rule commands are case-sensitive.

```
Rule {
    Process {
        Include OBJECT_NAME {
            -v "**"
        }
        Exclude .. {..}
    }
    Target {
        Match FILE {
            Include OBJECT_NAME {
                                -v "c:\\temp\\**"  #Specify file directory here
                        }
            Include -access "#define access type here"
                        Exclude -access "#define access type here"
        }
    }
}
```

To add more commands in Expert rules, see Learn Expert Rules commands for Files, Processes, and Registry.

Sections of Expert Rule syntax in detail

The above Expert rule syntax is described here:

`Rule`	Formulates the execution of commands defined within **Process** and **Target**.
`Process`	Executes the set of actions defined within the **Include** and **Exclude** commands. It does not take any other commands.

`Include OBJECT_NAME {`
`-v "**"`
`}`
`Exclude .. {..}`
	In this section,
	• The **Include** command considers the specified object name during file processing. When you specify **-v "**"**, all possible interfaces that users can

	interact with Windows are involved. To be more specific, you can write the object names such as **powershell.exe**, or **explorer.exe**, or **cmd.exe**. • The **Exclude** command eliminates the defined object name while processing. For more information, see Object name guidelines and Match types values.
`Target`	Defines the target matches for the rule. This command takes no arguments and can contain only **Match** commands. A rule must contain at least one or more **Target** commands.
`Match FILE`	Defines an object, that an Expert rule is intended to protect and to match an event. This command requires at lease one match object type value. For Files rule type, **FILE** is the match object type value.
`Include OBJECT_NAME { -v "c:\\temp\\**"`	In this section, you can define the target match type and the matching data. For example, **OBJECT_NAME** is used to specify the file directory and **-v** is specified to interpret the data as a single value.
`Include -access "#define access type here"` `Exclude -access "#define access type here"`	Defines the access flags. This flag applies when the protected file/file directory is accessed. The files rule type supports these access flags: • CONNECT_NAMED_PIPE • CREATE • DELETE • EXECUTE • POST • OPEN_FOR_DELETE • POST • READ • READ_DATA • RENAME • SET_REPARSE • WRITE • WRITE_ATTRIBUTE • WRITE_DATA

	To know more about the access flags, refer ACCESS_MASK flags.

For more Expert Rules examples, visit the Trellix Github repository.

Sample Expert Rules to protect files

Match Loaded_DLLs with AND/OR check

This Expert rule describes the usage of Loaded_DLL extension file along with the AND/OR matching on the loaded DLL files.

This Expert rule matches if "Test_DLL_Loaded.exe" has loaded "TestA.dll" AND "TestB.dll" AND "TestC.dll" AND ("TestD.dll" OR "TestE.dll"), then tries to launch "notepad.exe". The -xtype name must be unique as shown in the example. This is primarily useful in narrowing initiator matches.

Attention

Ensure to test this Expert rule on a client system before enforcing.

```
Rule  {
              Reaction BLOCK
                Process {
                                Include OBJECT_NAME { -v Test_DLL_Loaded.exe }
                                Include AggregateMatch -xtype "testa" {
                                                Include DLL_LOADED -name "testa" { -v 0x1 }
                                }
                                Include AggregateMatch -xtype "testb" {
                                                Include DLL_LOADED -name "testb" { -v 0x1 }
                                }
                                Include AggregateMatch -xtype "testc" {
                                                Include DLL_LOADED -name "testc" { -v 0x1 }
                                }
                                Include AggregateMatch -xtype "testd_or_teste" {
                                                Include DLL_LOADED -name "testd" { -v 0x1 }
                                                Include DLL_LOADED -name "teste" { -v 0x1 }
                                }
                }
                Target {
                                Match FILE {
                                                Include OBJECT_NAME { -v notepad.exe }
                                                Include -access "EXECUTE"
                                }
                }
}
```

For more Expert Rules examples, visit the Trellix Github repository.

Prevent file creation in a network path

This example rule prevents cmd.exe from creating files in a network path.

```
Rule {
        Process {
                Include OBJECT_NAME { -v cmd.exe  }
        }
        Target {
                Match FILE {
                        Include OBJECT_NAME { -v ** }
   Include -file_properties "FILE_NETWORK"
   Include -access "CREATE"
                }
        }
}
```

For more Expert Rules examples, visit the Trellix Github repository.

Prevent file creation

This example Expert rule triggers an event, when users create or access specific files in a file path, through Windows Command Prompt or File Explorer.

These are the file names and file path used in this example: **test.txt** and **test.dat c:\\temp**.

Attention

Make sure to test this Expert rule on a client system before enforcing wider.

```
Rule {
        Process {
                Include OBJECT_NAME {
                -v cmd.exe
                -v explorer.exe
                }
        }
        Target {
                Match FILE {
                        Include OBJECT_NAME { -v "c:\\temp\\*test.txt"}
                        Include -access "CREATE"
                }
                Match FILE {
                        Include OBJECT_NAME { -v "c:\\temp\\*test.dat"}
                        Include -access "CREATE WRITE READ"
                }
        }
}
```

Sections of Expert Rule syntax in detail

The above Expert rule is described here:

`Rule`	Formulates the execution of commands defined within **Process** and **Target**.
`Process`	Executes the set of actions defined within the **Include** and **Exclude** commands. It does not take any other commands.
`Include OBJECT_NAME { -v cmd.exe -v explorer.exe }`	In this section, **Include** command considers the defined object names in processing. As written in this rule, users are restricted to interact with Windows through the command prompt and File Explorer. For more information, see Object name guidelines and Match types values.
`Target`	Defines the target matches for the rule. This command takes no arguments and can contain only **Match** commands. A rule must contain at least one or more **Target** commands.
`Match FILE`	Defines an object, that an Expert rule is intended to protect and to match an event. This command requires at lease one match object type value. For Files rule type, **FILE** is the match object type value.
`Include OBJECT_NAME { -v "c:\\temp\ \*test.txt"}` `Include -access "CREATE"`	In this section, • **Include** command involves the file path defined within OBJECT_NAME. • **Include -access "CREATE"** blocks user to create **test.txt** in **c:\\temp** directory.
`Match FILE { Include OBJECT_NAME { -v "c:\ \temp\\*test.dat"} Include -access "CREATE WRITE READ" }`	As more than one file needs protection, subrules are defined in this Expert rule. Within the file directory **c:\\temp**, users cannot create, write, and read the file called **test.dat**.

For more Expert Rules examples, visit the Trellix Github repository.

Detect InstallUtil execution

Most of the malicious software uses InstallUtil to execute the untrusted files. This sample Expert rule detects when InstallUtil is used to execute .exe or .dll files.

You can exclude the known and trusted applications in the rule, to allow the regular processes for executing .exe and .dll files.

Attention

Make sure to test this Expert rule on a client system before enforcing wider.

```
Rule {
Process {
    Include DESCRIPTION { -v ".NET Framework installation utility" }
}
Target {
    Match FILE {
        Include OBJECT_NAME { -v "**.EXE**"  }
        Include OBJECT_NAME { -v "**.DLL**"  }

Exclude AggregateMatch {
        Include OBJECT_NAME { -v "C:\\WINDOWS\\**" }
        Include OBJECT_NAME { -v "C:\\PROGRAM FILES\\MCAFEE\\**" }
        Include OBJECT_NAME { -v "C:\\PROGRAM FILES\\COMMON FILES\\MCAFEE\\**" }
        Include OBJECT_NAME { -v "C:\\PROGRAM FILES (X86)\COMMON FILES\\MCAFEE\\**" }
        }

# Excluding known apps
        Exclude AggregateMatch {
            Include OBJECT_NAME { -v "**snake1.exe*" }
            Include MD5 { -v 2f3b994e836d731d04ad4cf0f37f10ab }
        }
        Include -access "READ WRITE CREATE EXECUTE"
        }
    }
}
```

For more Expert Rules examples, visit the Trellix Github repository.

Manage users from creating symbolic links and junctions

You can block non-privileged users or allow specific users from creating the symbolic links (symlinks) and junctions through **cmd.exe**, **powershell.exe**, or **powershell_ise.exe** by enforcing an Expert Rule using **ePO**.

Before you begin

- Identify the Security Identifiers(SID) groups or users that should be blocked or allowed to create symbolic links and junctions, in accordance with your corporate security policies. For more information about SID, refer Security identifiers on Microsoft's documentation.

- To allow the blocked users, ask them to run **whoami/groups** command in the Windows command prompt and know the SID groups they belong to.
- The Expert rule shown in this page is generic and covers the following permissions. For more information about security groups, refer Security identifiers on Microsoft's documentation.
    - System and High permissions — The groups that are allowed to run processes at Administrator permission level.
    - Medium and Low permissions — The groups that are allowed to run limited processes at Standard user and guest user level.

Task

1. **Select Menu → Policy → Policy Catalog, then select Endpoint Security Threat Prevention from the Products list in the left pane.**
2. **From the Category list in the right pane, select Exploit Prevention.**
3. **Click the Edit link for an editable policy.**
4. **Click Show Advanced.**
5. **In the Signatures section, click Add Expert Rule.**
6. **In the Expert Rules Properties page, specify the following fields.**
   ENS assigns the ID number for the rule automatically starting with 20000.
    a. **Enter Rule name.**
    b. **Select Severity.**
       Trellix recommends selecting **High** severity for initial validation.
    c. **Select Action.**
       Trellix recommends selecting **Report** action for initial validation. You can select Block and Report check boxes after validating that the rule works appropriately.
    d. **Select the Use Expert Rule template checkbox. This populates a template rule in the Rule content box based on the Rule type you select.**
    e. **Select Files in the Rule type drop-down list.**
    f. **Analyze which SID groups are appropriate to have permissions to create symbolic links and junctions, in accordance with your corporate security policies. Then, change the template code as shown here.**
        - To allow specific users or groups, add their SID within **Exclude AggregateMatch**.
        - To block specific users or groups, remove their SID or do not mention their SIDs within **Exclude AggregateMatch**.

Attention

Make sure to validate this Expert rule on a client test system before enforcing wider.

```
Rule {
    Process {
        Include OBJECT_NAME { -v cmd.exe }
        Include OBJECT_NAME { -v powershell.exe }
        Include OBJECT_NAME { -v powershell_ise.exe }

        # exclude admin groups
        Exclude AggregateMatch {
            Include GROUP_SID { -v "S-1-16-12288" }
            Include GROUP_SID { -v "S-1-16-16384" }
```

```
        }
    }
    Target {
        Match FILE {
            Include -access SET_REPARSE
        }
    }
}
```

In this rule, the High Mandatory Level (S-1-16-12288) and System Mandatory Level (S-1-16-16384) Security IDs are included within the **Exclude AggregateMatch** section. This blocks the non-privileged users and runs the process at administrative and system integrity level.

7. **Save the rule, then save the settings.**
8. **Validate the new Expert Rule on the client system.**
9. **Enforce the policy on a client system.**
   For more Expert Rules examples, visit the Trellix Github repository.

Expert Rules to protect processes

Create Expert Rules to protect processes using ePO

Expert rules can prevent endpoints from corrupting critical system processes and terminating security applications. You can create the Expert rules using ePO and protect processes by enforcing it in endpoints.

Task

1. **Select Menu → Policy → Policy Catalog, then select Endpoint Security Threat Prevention from the Products list in the left pane.**
2. **From the Category list in the right pane, select Exploit Prevention.**
3. **Click the Edit link for an editable policy.**
4. **Click Show Advanced.**
5. **In the Signatures section, click Add Expert Rule.**
6. **In the Expert Rules Properties page, complete the fields.**
   ENS assigns the ID number for the rule automatically starting with 20000.
   a. **In the Rule Name, provide a unique name for the Expert rule.**
   b. **Select the severity level according to the Expert rule.**
      The severity provides information only; it has no effect on the rule action.
   c. **Select Block and Report actions for the rule by selecting the corresponding check boxes.**
      Trellix recommends selecting **Report** action for initial validation. You can select **Block** and **Report** checkboxes after validating that the rule triggers the appropriate events.
   d. **Select the Use Expert Rule template checkbox. This populates a template rule in the Rule content box based on the Rule type you select.**
      To get a blank template for writing the Expert rules, deselect **Use Expert Rule template**.
   e. **Select Processes in the Rule type drop-down list.**
7. **Save the rule, then save the settings.**
8. **Validate the new policy on a client system.**

9. **Enforce the policy on the client systems.**

Create Expert Rules for processes on client system

You can create Expert rules directly on a client system or self-managed endpoints that aren't managed by ePO.

Before you begin

Make sure that the interface mode for the **Trellix Endpoint Security (ENS) Client** is set to Full access or log on to the **Trellix Endpoint Security (ENS) Client** as administrator.

Task

1. **Launch the Trellix Endpoint Security (ENS) Client.**
2. **Click Threat Prevention on the main Status page.**
   Or, from the **Action** menu, select **Settings**, then click **Threat Prevention** on the **Settings** page.
3. **Click Show Advanced.**
4. **In the Signatures section:**
   - Create a rule — Click **Add Expert Rule**.
   - Edit an existing user-defined rule — Double-click the rule in the table.
5. **In the Expert Rule Checker page, complete the fields.**
   Trellix ENS assigns the ID number automatically starting with 20000.
   a. **Select the severity and action for the rule.**
      The severity provides information only; it has no effect on the rule action.
   b. **Select Processes in the Rule Type drop-down list.**
   c. **Add the rule code to the Rule content field.**
6. **Save the rule, then save the settings.**
7. **Validate the new Expert Rule on the client system.**

Expert Rule syntax to protect processes

To write an expert rule to protect the Windows processes, you need to ensure that it follows the correct syntax. Expert rules consist of process, one or more targets, and the matching conditions that rules must examine before allowing the source to access the processes.

Here is a sample Expert rule for registry rule type and their respective definitions:

⚠ Caution

Expert Rule commands are case-sensitive.

```
Rule {
    Process {
        Include OBJECT_NAME {
            -v "**"
        }
    }
```

```
    Target {
        Match PROCESS {
            Include OBJECT_NAME {..}
            Include -access "DELETE TERMINATING" ; # Define access types
                   Exclude -access "CREATE"
        }
    }
}
```

To add more commands in Expert rules, see Learn Expert Rules commands for Files, Processes, and Registry.

Sections of Expert Rule syntax in detail

The above Expert rule syntax is described here:

`Rule`	Formulates the execution of commands defined within **Process** and **Target**.
`Process`	Executes the set of actions defined within the **Include** and **Exclude** commands. It does not take any other commands.

`Include OBJECT_NAME {`
`-v "**"`
`}`
`Exclude .. {..}`
	In this section,
	• The **Include** command involves the specified object name during processing. When you specify **-v "**"**, all possible interfaces that users/applications can interact with Windows are involved. To be more specific, you can write the object names such as **powershell.exe**, or **explorer.exe**, or **cmd.exe**.
	• The **Exclude** command eliminates the defined object name while processing.
	For more information, see Object name guidelines and Match types values.

`Target`	Defines the target matches for the rule. This command takes no arguments and can contain only **Match** commands. A rule must contain at least one or more **Target** commands.
`Match PROCESS`	Defines an object, that an Expert rule is intended to protect and to match an event. This command requires at lease one match object type value. Based on the object that needs protection, you use can one of these match object type values:

Match object_type_value	Description	Valid match object value
PROCESS	Controls access to an entire process handle.	• Initiator • Target  **Note:** If PROCESS is not used in the Initiator match, you must use THREAD.
SECTION	Controls access to creating a section object.	Target

			Note: If the access to be blocked is CREATE, the object type must be SECTION rather than PROCESS.
	THREAD	Controls access to a threat handle.	• Initiator • Target **Note:** If THREAD is not used in the Initiator match, you must use PROCESS.

`Include OBJECT_NAME {` `  -v notepad.exe`	This section defines the target object name, that is, Windows program to be secured.
`Include -access "DELETE TERMINATING"` `Exclude -access "CREATE"`	During processing, the **Include -access** and **Exclude-access** commands denote the access types for the specified program. You can write multiple access types together. For example, in this code snippet: • **Include -access "DELETE TERMINATING"** blocks the deletion and termination of object **notepad.exe**. • **Exclude -access "CREATE"** allows the creation of process or thread. The processes rule type supports these access types: • CREATE • DELETE • LOAD/IMAGE • TERMINATING • WRITE To know more about the access flags, refer ACCESS_MASK flags.

For more Expert Rules examples, visit the Trellix Github repository.

Sample Expert Rules to protect Processes

Prevent notepad execution

This Expert rule prevents users from executing notepad.exe through Windows File explorer. It also considers the size of the main module memory section.

Attention

Make sure to test this Expert rule on a client system before enforcing wider.

```
Rule {
    Process {
        Include OBJECT_NAME {
            -v explorer.exe
        }
    }
    Target {
        Match SECTION {
            Include OBJECT_NAME {
```

```
                -v "notepad.exe"
            }
            Include OBJECT_SIZE {
                -v 12345678
            }
            Include -access "EXECUTE" ; # Prevents section execution
        }
    }
}
```

When you validate this rule, execute the file **C:\Windows\notepad.exe**. This triggers an event in **Trellix Endpoint Security (ENS) Client**.

For more Expert Rules examples, visit the Trellix Github repository.

Block specific PowerShell parameters

This example rule prevents PowerShell from executing with specific command-line parameters, except for the encoded command, which is "dir c:\program files".

```
Rule {
        Process {
                Include OBJECT_NAME      { -v "*PowerShell*" }
                Include PROCESS_CMD_LINE { -v "*-NoLogo*" }
                Include PROCESS_CMD_LINE { -v "*-File*" }
                Include PROCESS_CMD_LINE { -v "*-EncodedCommand*" }
                Include PROCESS_CMD_LINE { -v "*-Command*" }
                Exclude PROCESS_CMD_LINE { -v "*-EncodedCommand
                                ZABpAHIAIAAnAGMAOgBcAHAAcgBvAGcAcgBhAG0AIABmAGkAbABlAHMAJwAgAA==" }
        }
        Target {
                Match SECTION { Include -access "CREATE" }
        }
}
```

For more Expert Rules examples, visit the Trellix Github repository.

Trigger a process scan

These examples show how to use the Reaction SCAN in the Expert Rule to trigger a process scan.

Example 1: Expert Rule with Reaction SCAN to scan actor process

When abc.exe launches xyz.exe, the Reaction SCAN ACTOR_PROCESS scans the actor process (abc.exe).

```
Rule {
    Reaction SCAN ACTOR_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
    Process {
        Include OBJECT_NAME { -v abc.exe}
    }
    Target {
        Match PROCESS {
            Include OBJECT_NAME { -v xyz.exe}
```

```
            Include -access "CREATE"
        }
    }
}
```

Example 2: Expert Rule with Reaction SCAN to scan target process

When abc.exe launches xyz.exe, the Reaction SCAN TARGET_PROCESS scans the target process (xyz.exe).

```
Rule {
    Reaction SCAN TARGET_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
    Process {
        Include OBJECT_NAME { -v abc.exe}
    }
    Target {
        Match PROCESS {
            Include OBJECT_NAME { -v xyz.exe}
            Include -access "CREATE"
        }
    }
}
```

Example 3: Expert Rule with Reaction SCAN to scan actor process when it accesses specific registry location

When the actor process (abc.exe) accesses a registry location that starts with HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options, it triggers a process scan of abc.exe.

```
Rule {
    Reaction SCAN ACTOR_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
    Process {
        Include OBJECT_NAME { -v abc.exe }
    }
    Target {
        Match KEY {
       Include OBJECT_NAME { -v "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File
Execution Options**"}
            Include -access "READ"
        }
    }
}
```

Example 4: Expert Rule with Reaction SCAN to scan actor and target process

When abc.exe launches xyz.exe, the Reaction SCAN ACTOR_PROCESS scans the actor process (abc.exe) and the Reaction SCAN TARGET_PROCESS scans the target process (xyz.exe).

```
Rule {
    Reaction SCAN ACTOR_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
    Reaction SCAN TARGET_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
    Process {
        Include OBJECT_NAME { -v abc.exe}
    }
    Target {
        Match PROCESS {
```

```
            Include OBJECT_NAME { -v xyz.exe}
            Include -access "CREATE"
        }
    }
}
```

Example 5: Chain rule (Next_Process_Behavior)

The Reaction SCAN command supports the Next_Process_Behavior chained rule ability. This Expert Rule shows that each Reaction SCAN command can have different scan actions.

```
Rule {
    Reaction SCAN ACTOR_PROCESS ScanAction REPORT
    Process {
        Include OBJECT_NAME { -v abc.exe }
    }
    Target {
        Match PROCESS {
            Include OBJECT_NAME { -v xyz.exe }
            Include -access "CREATE"
        }
        Next_Process_Behavior {
            Reaction SCAN ACTOR_PROCESS ScanAction REPORT_DELETE_PROCESS
            Reaction SCAN TARGET_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
            Target {
                Match PROCESS {
                    Include OBJECT_NAME { -v rmg.exe }
                    Include -access "CREATE"
                }
            }
        }
    }
}
```

For more Expert Rules examples, visit the Trellix Github repository.

Expert rules to protect Registry

Create Expert Rules to protect registry using ePO

Expert rule protects a specific registry by preventing users/unauthorized applications from accessing and modifying the registry keys or values in the registry. Based on the access permission you set in the rule, it blocks and triggers an event, if any unauthorized source access the protected registry.

Task

1. **Select Menu → Policy → Policy Catalog, then select Endpoint Security Threat Prevention from the Products list in the left pane.**
2. **From the Category list in the right pane, select Exploit Prevention.**
3. **Click the Edit link for an editable policy.**
4. **Click Show Advanced.**
5. **In the Signatures section, click Add Expert Rule.**

6. **In the Expert Rules Properties page, complete the fields.**
   Trellix ENS assigns the ID number for the rule automatically starting with 20000.
   a. **In the Rule Name, provide a unique name for the Expert rule.**
   b. **Select Block and Report actions for the rule by selecting the corresponding check boxes.**
      Trellix recommends selecting **Report** action for initial validation. You can select **Block** and **Report** check boxes after validating that the rule triggers the appropriate events.
   c. **Select the Severity level according to the Expert rule.**
      The severity provides information only; it has no effect on the rule action.
   d. **Select the Use Expert Rule template checkbox. This populates a template rule in the Rule content box based on the Rule type you select.**
      To get a blank template for writing the Expert rules, deselect **Use Expert Rule template**.
   e. **Select Registry in the Rule type drop-down list.**
7. **Save the rule, then save the settings.**
8. **Validate the new policy on a client system.**
9. **Enforce the policy on the client systems.**

Create Expert Rules for registry on client system

You can create Expert rules directly on a client system or self-managed endpoints that aren't managed by ePO.

Before you begin

Make sure that the interface mode for the **Trellix Endpoint Security (ENS) Client** is set to Full access or log on to the client system as administrator.

Task

1. **Launch the Trellix Endpoint Security (ENS) Client.**
2. **Click Threat Prevention on the main Status page.**
   Or, from the **Action** menu , select **Settings**, then click **Threat Prevention** on the **Settings** page.
3. **Click Show Advanced.**
4. **In the Signatures section:**
   - Create a rule — Click **Add Expert Rule**.
   - Edit an existing user-defined rule — Double-click the rule in the table.
5. **In the Expert Rule Checker page, complete the fields.**
   Trellix ENS assigns the ID number automatically starting with 20000.
   a. **Select the severity and action for the rule.**
      The severity provides information only; it has no effect on the rule action.
   b. **Select Registry in the Rule Type drop-down list.**
   c. **Add the rule code to the Rule content field.**
6. **Save the rule, then save the settings.**
7. **Validate the new Expert Rule on the client system.**

Expert Rule syntax to protect registry

To write an expert rule to protect your registry, you need to ensure that it follows the correct syntax. Expert rules consist of process, one or more targets, and the matching conditions that rules must examine before allowing the source to access the registry.

Here is a sample Expert rule for registry rule type and their respective definitions:

⚠ Caution

Expert Rule commands are case-sensitive.

```
Rule {
        Process {
                Include OBJECT_NAME {
                        -v regedit.exe
                }
        }
        Target {
                Match KEY {
                        Include OBJECT_NAME {
                                -v "HKLMS\\test**"
                        }
                Include -access "CREATE WRITE DELETE REPLACE_KEY RESTORE_KEY"
                }
        }
}
```

For more Expert rules examples, visit the GitHub repository.

To add more commands in Expert rules, see Learn Expert Rules commands for Files, Processes, and Registry.

Sections of Expert Rule syntax in detail

The above Expert rule syntax is described here:

`Rule`	Formulates the execution of commands defined within **Process** and **Target**.
`Process`	Executes the set of actions defined within the **Include** and **Exclude** commands. It does not take any other commands.
`Include OBJECT_NAME {` `-v regedit.exe`	**Include** command defines the Match types. These match types are supported for the Registry rule type: • ACCESS_MASK • AUTHENTICATION_ID

	• NT_ACCESS_MASK • OBJECT_NAME • OS_VERSION • REGVAL_DATA For example, the **OBJECT_NAME** match type is uses here to specify the name of interface object, regedit.exe, the Registry Editor. For more information, see Object name guidelines and Match types values.
`Target`	Defines the target matches for the rule. This command takes no arguments and can contain only **Match** commands. A rule must contain at least one or more **Target** commands.
`Match KEY`	Defines an object, that an Expert rule is intended to protect and to match an event. This command requires at lease one match object type value. For registry rule type, **KEY** and **VALUE** are the match object type values.
`Include OBJECT_NAME { -v "HKLMS\\test**"`	This section defines the target object name, that is, registry key path. These root keys are recognized: **Key** / **Matches** HKLM — HKLM is equivalent to HKEY_LOCAL_MACHINE. HKCU — All user registry keys (not just the current user) and the .default user key. HKCU is equivalent to: • HKEY_CURRENT_USER • HKEY_USERS

	Note: Matching against specific user SIDs is not supported.
HKCUC	All user classes (HKCU/*_CLASSES).
HKCR	System classes and all user classes (HKCU/*_CLASSES). HKCR is equivalent to HKEY_CLASSES_ROOT.
HKCCS	• HKLM/SYSTEM/CurrentControlSet • HKLM/SYSTEM/ControlSet00X
HKLMS	• HKLM/Software on 32-bit and 64-bit systems • HKLM/Software/Wow6432Node on 64-bit systems only
HKCUS	• HKCU/Software on 32-bit and 64-bit systems • HKCU/Software/Wow6432Node

	<table><tr><td></td><td>on 64-bit systems only</td></tr><tr><td>HKULM</td><td>• HKLM • HKCU</td></tr><tr><td>HKULMS</td><td>• HKLMS • HKCUS</td></tr><tr><td>HKALL</td><td>• HKLM • HKU</td></tr></table> **Note:** If the rule specifies a name where the root starts or contains a wild character, the AAC code performs no name normalization and that name might never match correctly. For example, ****\mcshield\start** is a valid name, but **H*L*\mcshield\start** is not. HKEY_CURRENT_CONFIG is not supported.
`Include -access "CREATE WRITE DELETE REPLACE_KEY RESTORE_KEY"`	Defines the access flags. This flag applies when the protected registry key or value is accessed. The registry rule type supports these access flags: • CREATE • DELETE • ENUM • LOAD_KEY • QUERY • READ • RENAME • REPLACE_KEY • RESTORE_KEY • WRITE To know more about the access flags, refer ACCESS_MASK flags.

For more Expert Rules examples, visit the Trellix Github repository.

Sample Expert Rules to protect registry

Prevent changing registry value

This Expert rule prevents changing the registry value available in the hive HKLMS\test through the Registry Editor.

Attention

Make sure to validate this rule in a client test system before enforcing it wider.

```
Rule {
        Process {
                Include OBJECT_NAME {
                        -v regedit.exe
                }
        }
Target {
        Match KEY {
                Include OBJECT_NAME {
                        -v "HKLMS\\test**"
                }
                Include -access "CREATE WRITE DELETE REPLACE_KEY RESTORE_KEY"
                }
        }
}
```

For more Expert Rules examples, visit the Trellix Github repository.

Prevent DLL injection through AppInit_DLLs

This Expert rule detects the untrusted process of injecting custom DLLs into the critical processes through AppInit_DLLs registry entry. When DLL adds into this registry entry, it forces user32.dll to load the DLL module during process startup.

Attention

Make sure to validate this rule in a client test system before enforcing it wider.

```
Rule {
    Process {
        Include VTP_TRUST true
    }
    Target {
        Match KEY {
            Include OBJECT_NAME {
                -v "HKLMS\\MICROSOFT\\WINDOWS NT\\CURRENTVERSION\\WINDOWS\\*_DLLs"
            }
            Include -access "CREATE WRITE DELETE REPLACE_KEY RESTORE_KEY"
        }
```

```
        Match VALUE {
            Include OBJECT_NAME {
                                -v "HKLMS\\MICROSOFT\\WINDOWS NT\\CURRENTVERSION\\WINDOWS\\*_DLLs"
            }
            Include -access "CREATE WRITE DELETE REPLACE_KEY RESTORE_KEY"
        }
    }
}
```

Sections of Expert Rule in detail

`Rule`	Formulates the execution of commands defined within **Process** and **Target**.
`Process`	Executes the set of actions defined within the **Include** and **Exclude** commands. It does not take any other commands.
`Include VTP_TRUST true`	Checks if VTP trusts the process or file. The value is treated as Boolean. That is, a value of 1 in the match type matches only processes trusted by VTP. A value of 0 matches non-trusted processes.
`Target`	Executes the **Match** command.
`Match KEY { Include OBJECT_NAME { -v "HKLMS\\MICROSOFT\\WINDOWS NT\ \CURRENTVERSION\\WINDOWS\\*_DLLs" } Include -access "CREATE WRITE DELETE REPLACE_KEY RESTORE_KEY"`	This section controls create, edit and delete access to key data in a key object, available in the folder, **HKLMS\\MICROSOFT\\WINDOWS NT\\CURRENTVERSION\ \WINDOWS*_DLLs**.
`Match VALUE { Include OBJECT_NAME { -v "HKLMS\\MICROSOFT\ \WINDOWS NT\\CURRENTVERSION\\WINDOWS\ \*_DLLs" } Include -access "CREATE WRITE DELETE REPLACE_KEY RESTORE_KEY"\\*_DLLs"`	This section controls create, edit and delete access to value data in a key object, available in the folder, **HKLMS\\MICROSOFT\\WINDOWS NT\\CURRENTVERSION\ \WINDOWS*_DLLs**.

The following actions can trigger events in the Event log page of ENS client:

- accessing the hive, **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows**

- creating a key, test.DLLs
- creating a DWORD value, test_DLLs
- creating a key, test_DLAs

For more Expert Rules examples, visit the Trellix Github repository.

Detect exporting SAM from registry

This Expert rule detects when unauthorized users/applications access Windows registry to export the Security Account Manager (SAM) database file.

Attention

Make sure to test this Expert rule on a client system before enforcing wider.

```
Rule {
    Process {
        Include AggregateMatch -xtype "1" {
            Exclude VTP_PRIVILEGES -type BITMASK { -v 0x8 }
        }
        Include AggregateMatch -xtype "2" {
            Exclude OBJECT_NAME { -v "TIWORKER.EXE" }
            Exclude OBJECT_NAME { -v "DEVICECENSUS.EXE" }
            Exclude OBJECT_NAME { -v "TRUSTEDINSTALLER.EXE" }
            Exclude OBJECT_NAME { -v "TASKHOSTW.EXE" }
            Exclude OBJECT_NAME { -v "OMADMCLIENT.EXE" }
            Exclude OBJECT_NAME { -v "SERVICES.EXE" }
            Exclude OBJECT_NAME { -v "CSRSS.EXE" }
            Exclude OBJECT_NAME { -v "SVCHOST.EXE" }
            Exclude OBJECT_NAME { -v "WINLOGON.EXE" }
            Exclude OBJECT_NAME { -v "SCHTASKS.EXE" }
            Exclude OBJECT_NAME { -v "REGEDIT.EXE" }
            Exclude OBJECT_NAME { -v "UpdateNotificationMgr.exe" }
            Exclude OBJECT_NAME { -v "**\\Program Files\\Common Files\\microsoft shared\\ClickToRun\
\*.exe" }
            Exclude OBJECT_NAME { -v "**\\Program Files (x86)\\Common Files\\microsoft shared\\ClickToRun\
\*.exe" }
            Exclude OBJECT_NAME { -v "**\\program files\\microsoft office\\**.exe" }
            Exclude OBJECT_NAME { -v "**\\program files (x86)\\microsoft office\\**.exe" }
            }
      }
      Target {
          Match KEY {
Include OBJECT_NAME { -v "HKLM\\SAM" }
Include OBJECT_NAME { -v "HKLM\\SAM\\Domain\\Account" }
Include OBJECT_NAME { -v "HKLM\\SECURITY\\Policy\\Secrets"}
Include -access "READ"
}
}
}
```

For more Expert Rules examples, visit the Trellix Github repository.

Expert rules to protect Buffer overflow

Create Expert Rules to prevent Buffer Overflow using ePO

You can create Expert rules to prevent buffer overflow exploits for the applications listed in the **Application Protection**. If not prevented, an attacker could use this vulnerability to execute custom hacking code on the machine and compromise security and data integrity.

Task

1. **Select Menu → Policy → Policy Catalog, then select Endpoint Security Threat Prevention from the Products list in the left pane.**
2. **From the Category list in the right pane, select Exploit Prevention.**
3. **Click the Edit link for an editable policy.**
4. **Click Show Advanced.**
5. **In the Signatures section, click Add Expert Rule.**
6. **In the Expert Rules Properties page, complete the fields.**
   ENS assigns the ID number for the rule automatically starting with 20000.
   a. **In the Rule Name, provide a unique name for the Expert rule.**
   b. **Select Block and Report actions for the rule by selecting the corresponding checkboxes.**
      Trellix recommends selecting **Report** action for initial validation. You can select **Block** and **Report** check boxes after validating that the rule works appropriately.
   c. **Select the Severity level according to the Expert rule.**
      The severity provides information only; it has no effect on the rule action.
   d. **Select the Use Expert Rule template checkbox. This populates a template rule in the Rule content box based on the Rule type you select.**
      To get a blank template for writing the Expert rules, deselect **Use Expert Rule template**.
   e. **Select Buffer Overflow in the Rule type drop-down list.**
7. **Save the rule, then save the settings.**
8. **Validate the new policy on a client system.**
9. **Enforce the policy on the client systems.**

Create Expert Rules for Buffer Overflow on client system

You can create Expert rules directly on a client system or self-managed endpoints that aren't managed by ePO.

Before you begin

Make sure that the interface mode for the **Trellix Endpoint Security (ENS) Client** is set to Full access or log on to the **Trellix Endpoint Security (ENS) Client** as administrator.

Task

1. **Launch the Trellix Endpoint Security (ENS) Client.**
2. **Click Threat Prevention on the main Status page.**
   Or, from the **Action** menu , select **Settings**, then click **Threat Prevention** on the **Settings** page.
3. **Click Show Advanced.**

4. **In the Signatures section:**
   - Create a rule — Click **Add Expert Rule**.
   - Edit an existing user-defined rule — Double-click the rule in the table.
5. **In the Expert Rule Checker page, complete the fields.**
   Trellix ENS assigns the ID number automatically starting with 20000.
   a. **Select the severity and action for the rule.**
      The severity provides information only; it has no effect on the rule action.
   b. **Select Buffer Overflow in the Rule Type drop-down list.**
   c. **Add the rule code to the Rule content field.**
6. **Save the rule, then save the settings.**
7. **Validate the new Expert Rule on the client system.**

Sample Expert Rule to prevent Buffer Overflow

To write an expert rule to prevent buffer overflow exploits, you need to ensure that it follows the correct syntax. This rule type is built based on the legacy **McAfee Host IPS**.

Here is a sample Expert rule for buffer overflow rule type and their respective definitions:

⚠ Caution

Expert Rule commands are case-sensitive.

```
Rule {
   time {Include "*"}
   application {Include "*"}
   user_name {Include "*"}
   attributes -no_trusted_apps -not_auditable
   directives "-d" "-c" "bo:stack" "bo:heap"
}
```

To add more commands in Expert rules, see Learn Expert Rules commands for Buffer overflow, Illegal API use and Services..

Sections of Expert Rule in detail

The above Expert rule is described here:

`Rule`	Formulates the execution of the defined commands.
`time {Include "*"}`	A section works based on the value mentioned in the **Include** keyword. For more information, see Include and Exclude keywords.

`application {Include "*"}`	Indicates that this rule is valid for all processes. To limit the rule to specific processes, list the pathname to each process.
`user_name {Include "*"}`	Indicates that this rule is valid for all users (or more precisely, the security context in which a process runs). To limit the rule to specific user contexts, list them using the form Local/user or Domain/user.
`attributes -no_trusted_apps -not_auditable`	Defines an object, that an Expert rule is intended to protect and to match an event. This command requires at lease one match object type value. For Files rule type, **FILE** is the match object type value.
`attributes -no_trusted_apps -not_auditable`	In this section, • **-no_trusted_apps** — Specifies that the trusted application list doesn't apply to this signature. • **-not_auditable** — Generates no exceptions for the signature when Adaptive mode is enabled.
`directives "-d" "-c" "bo:stack" "bo:heap"`	Directives add these behaviors: • **bo:stack** — Examines the memory location that is executing and detects if it is running from writable memory that is part of the current thread's stack. • **bo:heap** — Examines the memory location that is executing and detects if it is running from writable memory that is part of a heap. For more directives, see Buffer Overflow class type.

Expert rules to protect Illegal API use

Create Expert Rules to prevent Illegal API use using ePO

You can create Expert rules to prevent illegal use of the Exploit Prevention API. The Expert Rules can only extend the functionality of the Illegal API Use signatures provided by Exploit Prevention content. Expert Rules can't refer to APIs that aren't already covered in an Illegal API Use signature available in content.

Task

1. **Select Menu → Policy → Policy Catalog, then select Endpoint Security Threat Prevention from the Products list in the left pane.**
2. **From the Category list in the right pane, select Exploit Prevention.**
3. **Click the Edit link for an editable policy.**
4. **Click Show Advanced.**
5. **In the Signatures section, click Add Expert Rule.**
6. **In the Expert Rules Properties page, complete the fields.**
   ENS assigns the ID number for the rule automatically starting with 20000.
   a. **In the Rule Name, provide a unique name for the Expert rule.**
   b. **Select Block and Report actions for the rule by selecting the corresponding checkboxes.**
      Trellix recommends selecting **Report** action for initial validation. You can select **Block** and **Report** check boxes after validating that the rule works appropriately.
   c. **Select the Severity level according to the Expert rule.**
      The severity provides information only; it has no effect on the rule action.
   d. **Select the Use Expert Rule template checkbox. This populates a template rule in the Rule content box based on the Rule type you select.**
      To get a blank template for writing the Expert rules, deselect **Use Expert Rule template**.
   e. **Select Illegal API Use in the Rule type drop-down list.**
7. **Save the rule, then save the settings.**
8. **Validate the new policy on a client system.**
9. **Enforce the policy on the client systems.**

Create Expert Rules for Illegal API Use on client system

You can create Expert rules directly on a client system or self-managed endpoints that aren't managed by ePO.

Before you begin

Make sure that the interface mode for the **Trellix Endpoint Security (ENS) Client** is set to Full access or log on to the **Trellix Endpoint Security (ENS) Client** as administrator.

Task

1. **Launch the Trellix Endpoint Security (ENS) Client.**
2. **Click Threat Prevention on the main Status page.**
   Or, from the **Action** menu , select **Settings**, then click **Threat Prevention** on the **Settings** page.
3. **Click Show Advanced.**
4. **In the Signatures section:**
   - Create a rule — Click **Add Expert Rule**.
   - Edit an existing user-defined rule — Double-click the rule in the table.
5. **In the Expert Rule Checker page, complete the fields.**
   Trellix ENS assigns the ID number automatically starting with 20000.
   a. **Select the severity and action for the rule.**

The severity provides information only; it has no effect on the rule action.

b. **Select Illegal API Use in the Rule Type drop-down list.**

c. **Add the rule code to the Rule content field.**

6. **Save the rule, then save the settings.**
7. **Validate the new Expert Rule on the client system.**

Sample Expert Rule to prevent Illegal API Use

To write an expert rule to prevent illegal API use, you need to ensure that it follows the correct syntax. This rule type is built based on the legacy **Trellix Host IPS**.

Here is a sample Expert rule for illegal API use rule type:

Caution

Expert Rule commands are case-sensitive.

```
Rule {
        time {Include "*"}
        if { $EAGENT_64Bit_Process } {
                                        application {Include "[iEnv SystemRoot]\\system32\
\WindowsPowerShell\\*\\powershell.exe" \
                                                        "[iEnv SystemRoot]\\syswow64\\WindowsPowerShell\
\*\\powershell.exe"
                                                        }
                                    } else {
                                            application {Include "[iEnv SystemRoot]\\system32\
\WindowsPowerShell\\*\\powershell.exe" }
                                    }
        user_name {Include "*"}
        Vulnerability_Name {Include "Powershell Command Restriction - NoLogo"}
        directives "-d" "-c" "illegal_api_use:bad_parameter" "illegal_api_use:invalid_call"
        attributes -not_auditable
}
```

To know more about Expert rules commands, see Learn Expert Rules commands for Buffer overflow, Illegal API use and Services..

Expert rules to protect Services

Create Expert Rules to protect Services using Trellix ePO - On-prem

You can create Expert rules to protect the Windows Services (Windows versions 8.0 and earlier only). You can also create custom Services rules in the Access Protection policy in **Threat Prevention**. But, these rules don't provide the complete functionality available with Expert rules.

Task

1. **Select Menu → Policy → Policy Catalog, then select Endpoint Security Threat Prevention from the Products list in the left pane.**

2. **From the Category list in the right pane, select Exploit Prevention.**
3. **Click the Edit link for an editable policy.**
4. **Click Show Advanced.**
5. **In the Signatures section, click Add Expert Rule.**
6. **In the Expert Rules Properties page, complete the fields.**
   Trellix ENS assigns the ID number for the rule automatically starting with 20000.
   a. **In the Rule Name, provide a unique name for the Expert rule.**
   b. **Select Block and Report actions for the rule by selecting the corresponding checkboxes.**
      Trellix recommends selecting **Report** action for initial validation. You can select **Block** and **Report** check boxes after validating that the rule works appropriately.
   c. **Select the Severity level according to the Expert rule.**
      The severity provides information only; it has no effect on the rule action.
   d. **Select the Use Expert Rule template checkbox. This populates a template rule in the Rule content box based on the Rule type you select.**
      To get a blank template for writing the Expert rules, deselect **Use Expert Rule template**.
   e. **Select Services in the Rule type drop-down list.**
7. **Save the rule, then save the settings.**
8. **Validate the new policy on a client system.**
9. **Enforce the policy on the client systems.**

Create Expert Rules to protect Services on client system

You can create Expert rules directly on a client system or self-managed endpoints that aren't managed by ePO.

Before you begin

Make sure that the interface mode for the **Trellix Endpoint Security (ENS) Client** is set to Full access or log on to the **Trellix Endpoint Security (ENS) Client** as administrator.

Task

1. **Launch the Trellix Endpoint Security (ENS) Client.**
2. **Click Threat Prevention on the main Status page.**
   Or, from the **Action** menu, select **Settings**, then click **Threat Prevention** on the **Settings** page.
3. **Click Show Advanced.**
4. **In the Signatures section:**
   - Create a rule — Click **Add Expert Rule**.
   - Edit an existing user-defined rule — Double-click the rule in the table.
5. **In the Expert Rule Checker page, complete the fields.**
   Trellix ENS assigns the ID number automatically starting with 20000.
   a. **Select the severity and action for the rule.**
      The severity provides information only; it has no effect on the rule action.
   b. **Select Services in the Rule Type drop-down list.**
   c. **Add the rule code to the Rule content field.**
6. **Save the rule, then save the settings.**

7. **Validate the new Expert Rule on the client system.**

Sample Expert Rule to protect Services

To write an expert rule to protect Windows Services (Windows versions 8.0 and earlier only), you need to ensure that it follows the correct syntax. This rule type syntaxes are built based on the legacy **Trellix Host IPS**.

Here is a sample Expert rule for services rule type that prevents deactivation of Alerter service:

⚠ Caution

Expert Rule commands are case-sensitive.

```
Rule {
services { Include "Alerter" }
application { Include "*"}
user_name { Include "*" }
directives services:stop
}
```

To add more commands in Expert rules, see Learn Expert Rules commands for Buffer overflow, Illegal API use and Services..

Sections of Expert Rule in detail

The above Expert rule is described here:

`Rule`	Formulates the execution of the defined commands.
`services { Include "Alerter" }`	Indicates that the rule applies to the service with name **Alerter**. If the rule applies to multiple services, add them in this section in different lines.
`application {Include "*"}`	Indicates that this rule is valid for all processes. To limit the rule to specific processes, list the pathname to each process.
`user_name {Include "*"}`	Indicates that this rule is valid for all users (or more precisely, the security context in which a process runs). To limit the rule to specific user contexts, list them using the form Local/user or Domain/user.

`directives services:stop`	Indicates that this rule applies to deactivation of a service.

Validate and enforce an Expert Rule on a client system

Once you deploy a new Expert Rule to a client test system, validate that the syntax is correct and that it is working properly before deploying more widely. Validate that the syntax for an Expert Rule is correct and enforce it on a client test system to verify that it is working properly before deploying more widely. Syntax checking is available for Files, Registry, and Processes rule types only.

Before you begin

Make sure that the interface mode for the **Trellix Endpoint Security (ENS) Client** is set to **Full access** or log on to the **Trellix Endpoint Security (ENS) Client** as administrator.

Policy changes from **Trellix ePO - On-prem** might overwrite changes that you make to Expert Rules on the client system. Make sure to copy your changes back to the **Exploit Prevention** policy in the **Threat Prevention** module in **Trellix ePO - On-prem**.

Task

1. **Open the Trellix Endpoint Security (ENS) Client.**
2. **Click Threat Prevention on the main Status page.**
   Or, from the **Action** menu, select **Settings**, then click **Threat Prevention** on the **Settings** page.
3. **Click Show Advanced.**
4. **Click Exploit Prevention.**
5. **In the Signatures section, double-click a user-defined Expert Rule.**
6. **In the Expert Rule Checker window, click Check.**
   The **Check** button isn't available for Buffer Overflow, Illegal API Use, or Services rule types.
   If the syntax checker finds any errors:
   a. **Review the EndpointSecurityPlatform_errors.log file for information about the syntax error.**
   b. **In Trellix Endpoint Security (ENS) Client, correct the error.**
   c. **Click Check.**
   The **Enforce** button enables when the errors are resolved.
7. **Copy any updated Expert Rules to the Exploit Prevention policy in the Threat Prevention module in Trellix ePO - On-prem.**
8. **Click Enforce to save and enforce the rule or Close to cancel any changes and close the Expert Rule Checker window.**
9. **Perform the restricted actions that you have written in the rule.**
10. **Navigate back to the Event Log page in ENS.**
    You can view the events that are triggered for violating the rule. If intended action is not reported, make sure that you have selected **Report** check box while creating or enforcing Expert rules.

Learn Expert Rules for files, processes, and registry

AAC rule structure

Rules define the boundaries of acceptable behavior and tell AAC how to react when the filtered action matches the rule specifications.

The **Rule** command at the root level defines the rule. Each Expert Rule identifier can contain only one rule definition and multiple subrules. The **Match** command defines subrules, each of which has an assigned role: **Initiator** or **Target**.

Because **Initiator** subrules always apply to PROCESS objects, the **Process** command provides a shortcut method for defining **Initiator** sections.

Note

Commands for building AAC rules are case sensitive.

Here is the basic structure of AAC-based rules:

```
Rule {
        Initiator {
                Match … {
                        Include … { … }
                        Exclude … { … }
                }
        }
        Target {
                Match … {
                        Include … { … }
                        Exclude … { … }
                }
        }
}
```

Note

Trellix ENS doesn't support signatures with multiple rules.

Expert Rule commands

Rule command

The **Rule** command defines an AAC rule. Each Expert Rule identifier can contain only one rule definition.

Description

This command takes no arguments and can contain one or more **Initiator**, **Process**, and **Target** commands. Only the **Target** command is required.

Syntax

```
Rule    {
        Initiator …
        Process …
        Target …
}
```

Initiator command

The **Initiator** command in a **Rule** command defines the AAC initiator matches. Only processes can be initiators.

Description

This command takes no arguments and can contain only **Match** commands.

A **Rule** command must contain at least one Initiator command and can contain multiple **Initiator** commands. If the value isn't specified, the rule uses **** to indicate all processes.

Syntax

```
Rule {
        …
        Initiator       {
                Match …
        }
        …
}
```

Process command

The **Process** command provides a shortcut method for defining **Initiator** **Match** sections.

Description

This command takes no arguments and can contain multiple **Include** and **Exclude** commands.

A **Rule** command can contain multiple **Process** commands. The **Process** command is optional. If not specified, the rule uses the value **** to indicate all processes.

Syntax

```
Rule {  ...
        Process {
        }
        ...
}
```

This syntax is a shortcut for:

```
Rule    { ...
        Initiator {
```

```
                Match PROCESS {
                }
        }
        ...
}
```

Target command

The **Target** command defines the AAC target matches for the rule.

Description

This command takes no arguments and can contain only **Match** commands.

A **Rule** must contain at least one **Target** command and can contain multiple **Target** commands.

Syntax

```
Rule {
        Target  {
                Match …
        }
 ...
}
```

Next_Process_Behavior command

The **Next_Process_Behavior** command defines a new chained link in the AAC target. **You can use this command to create behavioral rules to block a specific sequence of actions.**

Description

This command takes no arguments and can contain only **Target** commands.

A **Rule** command can contain multiple chained links definitions. Each **Next_Process_Behavior** command must be defined together with a **Match** command with the PROCESS *object_type_value* within a **Target** command.

Syntax

```
Rule { ...
 Target {
   Match PROCESS {
     ...
   }
   Next_Process_Behavior {
     Target {
       ...
     }
   }
 ...
 }
}
```

Match command

The **Match** command defines the criteria that AAC uses to match an event.

Description

This command takes one required argument, *object_type_value*, which specifies the case-sensitive AAC object type to match, and can contain multiple **Include** and **Exclude** commands.

The **Match** command can be used in **Initiator** and **Target** commands only.

Syntax

```
Rule    {
        Initiator
                        Match object_type_value {
                                Include …
                                Exclude …
                        }
        }
        Target
                        Match object_type_value {
                                Include …
                                Exclude …
                        }
        }
}
```

Include and Exclude commands

The **Include** and **Exclude** commands specify the data used for matching.

Description

The **Include** and **Exclude** commands take two required arguments:

- *MATCH_type*, which determines the entries in an **Include** or **Exclude** that are ORed or ANDed
- The actual data to match The body of the command can contain multiple data entries. Each data entry must begin with either **-v** or **-l**.

Syntax

```
Rule    {
        Initiator
                Match   {
                        Include MATCH_type < -type PATH > {
                                -v data | -l data
                                ...
                        }
                }
        }
}
```

```
Rule    {
        Initiator
                Match   {
                        Exclude MATCH_type < -type PATH > {
                                -v data | -l data
                                ...
                        }
                }
        }
}
```

Arguments

Argument	Description
-v	Specifies to interpret the following entry as a single value.
-l	Specifies to interpret the following entry as a Tcl list — each entry in the list is automatically broken out into its own match entry.
-pfx	Specifies a string to prepend to all following data entries. The strings remain in effect until another **-pfx** option. To remove the current value, use this option with no string.
-sfx	Specifies a string to append to all following data entries. The strings remain in effect until another **-sfx** option. To remove the current value, use this option with no string.
-type PATH	Treats all entries in the body as paths and automatically removes any trailing directory separators: **/** or ****. This is useful to avoid double separators when you are appending strings to the values with the **-sfx** option.

Shortcuts for MATCH_type

You can use the following shortcuts instead of building the entire *MATCH_type* entry.

Syntax

```
Include/Exclude -processor_mode  user|kernel
```

```
Include/Exclude -vtp_trust  true|false
```

```
Include/Exclude -access access_types
```

The *access_types* value is a list of access tokens separated by a delimiter and is case insensitive. The valid delimiters are a space, tab, comma, or pipe |.

The valid access tokens are:

- CLEANUP
- CLOSE
- CONNECT_NAMED_PIPE
- CREATE
- DELETE
- ENUM
- EXECUTE
- LOAD_IMAGE
- LOAD_KEY
- OBJECT_EXISTS
- OPEN_NAMEDSECTION
- POST
- QUERY
- READ
- REPLACE_KEY
- RESTORE_KEY
- SET_REPARSE
- SET_SECURITY
- START_DEVICE
- TERMINATING
- WRITE
- WRITE_ATTRIBUTE

AggregateMatch command

The **AggregateMatch** command defines a list of data that AAC uses to match an event. **You can use this command to create a list of values to match in a rule so you can use the same data without having to rewrite the values.**

Description

This command takes no arguments and can be used in **Include** and **Exclude** commands only.

The *Match_type* value is required for each item in **AggregateMatch**.

Syntax

```
Rule    {
 Initiator {
   Match object_type_value {
     Include AggregateMatch {
       Include ...
       Exclude ...
     }
     Exclude AggregateMatch {
       Include ...
       Exclude ...
     }
   }
 }
  ...
 Target {
   Match object_type_value {
     Include AggregateMatch {
       Include ...
       Exclude ...
     }
     Exclude AggregateMatch {
       Include ...
       Exclude ...
     }
   }
 }
 ...
}
```

Reaction SCAN command

The Reaction SCAN command defines the ability to perform process scans when a rule matches.

Description

Note

The Reaction SCAN command is available with **Trellix ENS** 10.7 November 2020 Update and later.

This command takes two arguments:

- Process to be scanned (ACTOR_PROCESS and/or TARGET_PROCESS).
- ScanAction, the action to take when a detection occurs.

Syntax

An example of a rule with the Reaction SCAN command. The command precedes the Process clause in this example of a simple Expert Rule where the Reaction SCAN command is used.

```
Rule {
    Reaction SCAN ACTOR_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
    Reaction SCAN TARGET_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
    Process {
        Include OBJECT_NAME { -v abc.exe}
    }
    Target {
        Match PROCESS {
            Include OBJECT_NAME { -v xyz.exe}
            Include -access "CREATE"
        }
   }
}
```

The Reaction SCAN command can also be used in chained Expert Rules. The command is placed at the top of the Next_Process_Behavior clause.

```
Rule {
    Reaction SCAN ACTOR_PROCESS ScanAction REPORT
    Process {
        Include OBJECT_NAME { -v abc.exe }
    }
    Target {
        Match PROCESS {
            Include OBJECT_NAME { -v xyz.exe }
            Include -access "CREATE"
        }
        Next_Process_Behavior {
                Reaction SCAN TARGET_PROCESS ScanAction REPORT
            Target {
                Match PROCESS {
                    Include OBJECT_NAME { -v rmg.exe }
                    Include -access "CREATE"
                }
            }
        }
    }
  }
```

Example

```
Reaction SCAN ACTOR_PROCESS ScanAction REPORT_CLEAN_DELETE_PROCESS
```

When the Expert Rule matches, the actor process is scanned with a scan action that tries to clean the process. If the clean action is not successful, it attempts to delete the process and will log the detection to On-Demand Scan Activity log and report the detection to **Trellix ePO - On-prem**.

```
Reaction SCAN TARGET_PROCESS ScanAction REPORT
```

When the Expert Rule matches, the target process is scanned with a scan action that will log the detection to On-Demand Scan Activity log and report the detection to **Trellix ePO - On-prem**.

For more detailed examples on how to use Reaction SCAN, see Expert rule triggered process scan.

Note

The process scan only works on processes, it does not work on files and registry. The process scan takes around 2 seconds to complete; on a detection the complete scan takes between 2–9 seconds to fix it.

Scan actions

These are the scan actions and their descriptions.

Scan action	Description
CLEAN_PROCESS	Attempts to clean the process. The detection is logged to the On-Demand Scan Activity log.
DELETE_PROCESS	Attempts to delete the process. The detection is logged to the On-Demand Scan Activity log.
CLEAN_DELETE_PROCESS	First attempts to clean the process, if unsuccessful then attempt to delete the process. The detection is logged to the On-Demand Scan Activity log.
REPORT	No action is taken on the detected process. The detection is logged to the On-Demand Scan Activity log and a detection event is sent to **Trellix ePO - On-prem**.
REPORT_CLEAN_PROCESS	Attempts to clean the process. The detection is logged to the On-Demand Scan Activity log and a detection event is sent to **Trellix ePO - On-prem**.
REPORT_DELETE_PROCESS	Attempts to delete the process. The detection is logged to the On-Demand Scan Activity log and a detection event is sent to **Trellix ePO - On-prem**.

Scan action	Description
REPORT_CLEAN_DELETE_PROCESS	First attempts to clean the process, if unsuccessful then attempt to delete the process. The detection is logged to the On-Demand Scan Activity log and a detection event is sent to**Trellix ePO - On-prem**.

If multiple Reaction SCAN commands are included in an Expert Rule, each command can have a different scan action.

Task-less process scan

When the Expert Rule matches, a process scan request is sent asynchronously to On-Demand Scan. The On-Demand Scan performs the process scan as a Task-less On-Demand Scan. This scan happens immediately and does not depend on the scheduled scan task.

Events and log details

Detection logging

If a detection occurs, the detection is logged in the On-Demand Scan Activity log. The detection information in the log includes the rule ID, rule name which triggered the scan, the name of the detected process, and the scan action (remediation action) taken.

Trellix ePO - On-prem events

If one of the report scan actions (REPORT, REPORT_CLEAN_PROCESS, REPORT_DELETE_PROCESS, REPORT_CLEAN_DELETE_PROCESS REPORT) is used, a detection event is sent to **Trellix ePO - On-prem**. The detection event follows the same format as of On-Demand Scan process scan detection events except for the fields **Task Name**, **Analyzer Rule ID**, and **Analyzer Rule Name**.

These fields are named in these format:

- **Expert Rule On-Demand Process Scan**
- **Expert Rule ID**
- **Expert Rule Name**

Scanned or not scanned log details

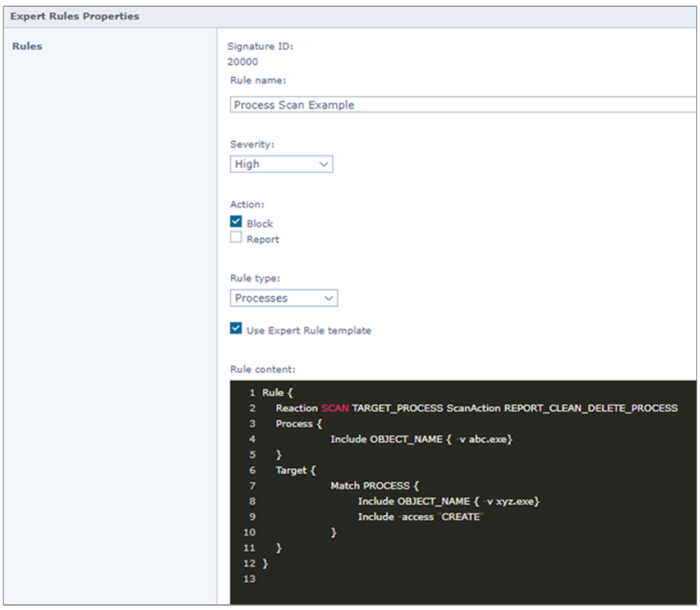
Each scan is logged to the On-Demand Scan Debug log. If a scan does not occur for some reason, the rule ID, rule name, process name, and the reason why the scan could not be completed is logged in the On-Demand Scan Activity log.

Debug logging

The complete flow of an Expert Rule triggering a process scan can be seen by enabling debug logging for Exploit Prevention and On-Demand Scan. The Expert Rule sending the scan request is seen in the Exploit Prevention Debug log, and the handling of the process scan is seen in On-Demand Scan Debug log.

Special consideration

- The process scan is not able to scan Windows protected process because it can't get access to the process memory. The inability to scan a process due to this is logged in the On-Demand Activity log, citing that the process scan is unable to access the process for scanning.
- It is possible to create Expert Rules which match thousands of times a second, and if the Expert Rule includes the Reaction SCAN command then, if unchecked, could request a process scan thousands of times a second. Therefore safeguards are in place that will limit one process scan, per process at a time.
- Expert Rule action can be executed to block and report. In the given example, the Expert Rule is configured to block creating xyz.exe and it also has a reaction to scan the target process, xyz.exe. In this case, the process scan does not occur because the process launch is blocked. The attempt to perform process scan on xyz.exe fails because the scanner is not able to access the xyz.exe process.



How match criteria in AAC-based subrules are evaluated

The match criteria in each subrule specifies either the **Include** or **Exclude** directive. The rule engine evaluates the filtered event against the match criteria in the subrule.

The subrule matches the filtered event if both of the following are true:

- At least one **Initiator** subrule matches the process that initiated the action described by the event.
- At least one **Target** subrule matches the object type that is the subject of the action.

When evaluating a filtered event against a subrule, the rule engine performs logical OR between matching criteria of the same type and logical AND between matches of different type. The rule engine first evaluates the matches with the **Exclude** directive, and then evaluates the matches with the **Include** directive.

The subrule evaluates to TRUE if both of the following are true:

- Exclude matches evaluate to FALSE.
- Include matches evaluate to TRUE.

Example

```
Rule {
        Initiator {
                Match PROCESS {
                                Include OBJECT_TYPE_A      { … }
                                Include OBJECT_TYPE_B      { condition 1 }
                                Include OBJECT_TYPE_B      { condition 2 }
                                Exclude OBJECT_TYPE_C      { … }
                        }
                Target {
                        Include OBJECT_TYPE_D     { condition 1 }
                        Include OBJECT_TYPE_D     { condition 2 }
                }
}
```

This rule evaluates to TRUE if both the following are TRUE:

- One of the following **Initiator** conditions is TRUE:
  - OBJECT_TYPE_A and OBJECT_TYPE_B condition 1 are TRUE.
  - OBJECT_TYPE_A and OBJECT_TYPE_B condition 2 are TRUE.
  - OBJECT_TYPE_A is TRUE and OBJECT_TYPE_C is FALSE.
- One of the following **Target** conditions is TRUE:
  - OBJECT_TYPE_D condition 1 is TRUE.
  - OBJECT_TYPE_D condition 2 is TRUE.

Valid parent-child relationships between AAC commands

The AAC syntax defines which commands can be the parent or children of other commands.

Command	Parent	Children
Rule	Not applicable	Initiator
		Process
		Target
Initiator	Rule	Match
Process	Rule	Include
		Exclude

Command	Parent	Children
Target	Rule	Match
Match	Initiator	Include
	Target	Exclude
Include	Process	Not applicable
	Match	
Exclude	Process	Not applicable
	Match	

Match object type values

The **Match** command takes one required argument, *object_type_value*, which is the case-sensitive AAC object type to match.

This table lists the valid values of *object_type_value*.

Match object_type_ value	Description	Valid match object type	Notes
FILE	Controls access to a file.	Target	
KEY	Controls access to both key and value data in a key object.	Target	
PROCESS	Controls access to a process handle.	• Initiator • Target	If PROCESS is not used in the Initiator match, you must use THREAD. If the access to be blocked is CREATE, the object type must be SECTION rather than PROCESS.

Match object_type_ value	Description	Valid match object type	Notes
SECTION	Controls access to creating a section object.	Target	
THREAD	Controls access to a thread handle.	• Initiator • Target	If THREAD is not used in the Initiator match, you must use PROCESS.
VALUE	Controls access to value data in a key object.	Target	

Match type values

The *MATCH_type* value determines which entries in an **Include** or **Exclude** are ORed or ANDed. Commands with the same *MATCH_type* value evaluate to either value (OR). Commands with different *MATCH_type* values evaluate to both values (AND).

Each *Match_type* value uses a specific data type for its possible values. The supported data types are:

- **INT*x*/UINT*x*** — All match 32-bit or 64-bit numeric values.
- **STRING** — A null-terminated text string.
- **BITMASK** — A numeric value expressed in hexadecimal notation, which is logically evaluated, such as 0xfe340ead.
- **BINARY** — Binary data specified as a hexadecimal value, such as fe340ead.
- **MULTI_STRING** — Sequence of null-terminated strings that are terminated by two null characters, such as "string1\0string2\0string3\0\0".
- **EXPANDABLE_STRING** — A null-terminated string that contains unexpanded references to environment variables, such as "%PATH%".

Note

MATCH_types values are case sensitive.

Match type value	Description	Data type	Valid in object types
ACCESS_MASK	Specifies the access type.	UINT64 - BITMASK	All

Match type value	Description	Data type	Valid in object types
AUTHENTICATION_ID	Matches a textual account SDDL SID identifier. This match can be used to identify a specific user-account in policy enforcement.	STRING	All
CACHE_ATTRIBUTE	Matches a cache attribute for the given object. Because it is a bitmask match type, any matching bits are considered a match.	BITMASK	• FILE • PROCESS
CERT_HASH	Matches the certificate hash; doesn't check whether the cert is chained to the root. If the object is of type PROCESS or THREAD, the certificate is obtained from the main entry module. This match never evaluates to true if the object is not signed.	UINT8[16]	• PROCESS • SECTION • THREAD
CERT_NAME	Matches the object's signing certificate name, but doesn't check whether the certificate is chained to the root. If the object is of type PROCESS or THREAD, the certificate is obtained from the main entry module.	STRING	• PROCESS • SECTION • THREAD

Match type value	Description	Data type	Valid in object types
	This match never evaluates to true if the object is not signed.		
CERT_NAME_CHAINED	Matches the object's signing certificate name, and the signing certificate must be chained to the root of the certificate store. If the object is of type PROCESS or THREAD, the certificate is obtained from the main entry module. This match never evaluates to true if the object is not signed.	STRING	• PROCESS • SECTION • THREAD
DESCRIPTION	Matches the “FileDescription” resource extracted from the resource section for the PE.	STRING	• FILE • PROCESS • SECTION
DISKIO_HOOK	Specifies the source for the filtered disk IO, where **upper** designates IO origin to be the upper disk filter, mfedisk.sys and **miniport** and **firmware** refer to IOs originating from the CAPI library, mfecapik.sys.	UINT32	DISK
DISK_REGION	Specifies the accessed disk region types as defined by AAC. This matching criteria aids	UINT64 - BITMASK	DISK

Match type value	Description	Data type	Valid in object types
	in creating minimal rules protecting special/ interesting disk areas. The defined bits are: • MBR — Matches access to the MBR (Master Boot Record) and the subsequent 63 sectors. For GPT-style disks, this bit also applies to accesses to the partition table (LBAs 1–33 inclusive), including the mirror table at the end of the disk. • VBR — Matches access to the VBR (Volume Boot Record). • PARTITION — The accessed LBAs are in a single partition. • NOT_PARTITIONED — One or more of the accessed LBAs are in an area that is not partitioned. This bit always matches access to RAW/uninitialized disks. • MULTI_PARTITION — The accessed LBAs span more than one partition.		
DLL_LOADED	Matches a loaded DLL in a specified PROCESS object.	BITMASK	PROCESS

Match type value	Description	Data type	Valid in object types
	This is primarily useful for narrowing **Initiator** matches, such as svchost.exe service exclusions. The DLL name generally is the base name of the DLL without a path or file extension. That is, “MFEVTPA” matches, whereas “MFEVTPA.DLL” or “c:\program files\common files\mcafee\systemcore\mfevtpa.dll”. The match data is pulled directly from the process structures where the DLL is known by its base name and the associated image file name is not present. To match when the DLL is loaded, set the value part of the name-value bitmask to 1. To match when the DLL is not loaded, set it to 0.		
ENV_VAR	Specifies an environment variable name and its value. This criteria matches only if both name and value match the environment variables extracted from the PEB.	Named value pair: STRING, STRING	• PROCESS • THREAD

Match type value	Description	Data type	Valid in object types
EXP_USER_NAME	Selects the local account SID, when an authenticating authority isn't defined in the rule.	STRING	• FILE • PROCESS
FILE_ATIME	Matches against the file last accessed time.	INT64	• FILE • PROCESS
FILE_ATTRIBUTES	Matches against the file attribute bits.	BITMASK	• FILE • PROCESS
FILE_CTIME	Matches against the file create time.	INT64	• FILE • PROCESS
FILE_MTIME	Matches against the file last changed time.	INT64	• FILE • PROCESS
FILE_PROPERTIES	Matches the bitmask against file properties reported by the **Target**. The defined bits are: • NETWORK (0x1) — File is in a network path. • REMOVABLE (0x2) — File is on a removable drive. • FLOPPY (0x4) — File is on a floppy drive. • CD (0x8) — File is on a CD drive. • DFS (0x10) — File is over on DFS. • REDIRECTOR (0x20) — File is opened using a redirector.	UINT64 - BITMASK	FILE

Match type value	Description	Data type	Valid in object types
GROUP_NAME	Matches the provided textual name against the groups that the user token belongs to. The criteria evaluates to true if at least one matching group is found.	STRING	• PROCESS • THREAD
GROUP_SID	Matches the provided textual SID (that is, S-1-5-18) against the groups that the user token belongs to. The criteria evaluates to true if at least one matching group is found.	STRING	• PROCESS • THREAD
IMAGE_BASE_ADDRESS	Specifies the virtual base address for an image. This is useful for retrieving the base address for an image during an image load notification.	UINT64	SECTION Available only during load image callbacks, access mask set to LOAD_IMAGE.
IMAGE_ENTRY_POINT	Specifies the entry point offset (in bytes) for an image. This is useful for retrieving the entry point address for an image during an image load notification.	UINT64	SECTION Available only during load image callbacks, access mask set to LOAD_IMAGE.
IMAGE_PROPERTIES	Specifies different image properties, as	UINT64 - BITMASK	SECTION

Match type value	Description	Data type	Valid in object types
	available during an image load notification. The defined bits are: • 64-bit — 64-bit image. • SYSTEM_MODE — System mode image. • MAPPED_TO_ALL_PROCESSES — The image is mapped to all processes.		Available only during load image callbacks, access mask set to LOAD_IMAGE.
IS_DIRECTORY	Matches operations against files or directories: • 0 — files • 1 — directories	UINT8 - Boolean	FILE
IS_TRANSACTED	Matches (true) if the file is part of an NTFS TxF transaction. For PROCESS or THREAD object types, matches if the backing file object for the main executable is part of an NTFS TxF transaction.	UINT8 - Boolean	• FILE • PROCESS • SECTION • THREAD
KERNEL_CALLER_NAME	Matches the name of the kernel module that issued the disk IO.	STRING	DISK Valid only in the context of DeepStore.
LBA	Compares the specified LBA (Logical Block Address) to the one that is being accessed. The location of the MBR (Master Boot Record) is always LBA 0.	UINT64	DISK

Match type value	Description	Data type	Valid in object types
LBA2FILE	While filtering disk I/O, matches the specified name against the name, according to the CAPI content driver, of the file in the filtered LBA.	STRING	DISK
LBA_FROM_END	Calculates the accessed LBA using a reverse scheme in which the last sector on the disk is considered to carry LBA 0. For example, match data that specifies range 0..1 matches access to the last 2 sectors. On a disk with N+1 blocks/sectors, where LBAN is the last block (using a 0-based scheme), match data 1..2 corresponds to access to LBAN-2 and LBAN-1. This criterion is provided for convenience, so that rules can protect several sectors, starting from an offset calculated from the end of the disk, without knowing the disk size.	UINT64	DISK
MD5	Indicates the MD5 digest of the backing file. If object is of type PROCESS or THREAD,	UINT8	• FILE • PROCESS • SECTION

Match type value	Description	Data type	Valid in object types
	MD5 is calculated against its main executable module.		• THREAD
NT_ACCESS_MASK	Matches against the native NT access mask of the I/O operation for file, registry, process, and thread access attempts. Make sure to use access masks appropriate for the object type as described in Microsoft MSDN. For example, to use NT_ACCESS_MASK to block calls to CreateFile() with GENERIC_WRITE, the bit mask must be FILE_GENERIC_WRITE. **Note:** Due to operating system limitations, you can't block PROCESS_QUERY_LIMITED_INFORMATION but you can use it in ALLOW rules for reporting purposes.	UINT64 - BITMASK	• FILE • PROCESS • REGISTRY • THREAD
OBJECT_NAME	Specifies the object name. Any combination of wildcards is accepted.	STRING	All

Match type value	Description	Data type	Valid in object types
OBJECT_SIZE	Matches against the size of the file or, for a section, the image size during load.	INT64	• FILE • SECTION
OPERATION_STATUS	Matches the operation status for a post-event. Not useful with non-post events.	INT32	FILE
OS_VERSION	Compares the specified operating system version to the actual version. The operating system version must be specified in the format: **OS_Version = Major_Version * 1000 + Minor_Version * 10 + ServicePack. By way of example: VistaRtm = 6000; VistaSp1=6001; Win7=6010; Win7Sp1=6011; Win8=6020**	UINT32	All
PARTITION_STYLE	Compares the match criteria with the partition style of the disk **Target**.	UINT32	DISK
PE	Matches a data value of "1" if the target file is a PE (Portable Executable, Windows executable binary) file.	UINT8	FILE

Match type value	Description	Data type	Valid in object types
	Note: Initiator PROCESS/THREAD matches are not supported because, by definition, they are PE files.		
PE_MD5	Compares MD5 digest calculated across PE against the match criteria. The digest is calculated according to Microsoft Authenticode PE hash value calculations – 4-byte PE header check sum is omitted as well as the Certificate Table Entry, which is part of Optional Header Directories.	UINT8	• FILE • PROCESS • SECTION • THREAD
PE_SHA1	Compares the match data with the SHA-1 hash sum calculated across the PE.	UINT8	• FILE • PROCESS • SECTION • THREAD
PE_SHA2_256	Compares the match data with the SHA2-256 hash sum calculated across the PE.	UINT8	• FILE • PROCESS • SECTION • THREAD
PE_SHA2_384	Compares the match data with the SHA2-384 hash sum calculated across the PE.	UINT8	• FILE • PROCESS • SECTION • THREAD

Match type value	Description	Data type	Valid in object types
PE_SHA2_512	Compares the match data with the SHA2-512 hash sum calculated across the PE.	UINT8	• FILE • PROCESS • SECTION • THREAD
PROCESSOR_MODE	Matches if the match is evaluated in the context of an I/O operation originating from user-mode or kernel-mode. This is most useful for excluding processes from matching a rule if the process is executing in user-mode. **Note:** Do not use this type with registry operations.	UINT8 - KPROCESSOR_MODE (0 = kernelmode, 1 = usermode)	• PROCESS • THREAD
PROCESS_CMD_LINE	Matches the process command line, extracted from the PEB (Process Environment Block), a data structure used by Microsoft Windows to hold information about running processes.	STRING	• PROCESS • THREAD
PROCESS_ID/ THREAD_ID	Matches a specified thread ID.	UINT64 - Thread ID	• PROCESS • THREAD

Match type value	Description	Data type	Valid in object types
	Note: Remember when using this match type that thread IDs and process IDs are rapidly recycled in the Windows environment.		
PROCESS_STATE_BITS	Compares the specified name/bitmask with the stateID/stateBits carried by the **Initiator** or **Target** ProcessInfo object. The comparison evaluates to true if stateBits with stateID are present in ProcessInfo and the "bitwise and" between the stateBits and the bitmask carried by the match object yields a non-zero result.	BITMASK	• PROCESS • THREAD
PRODUCT_NAME	Matches the "ProductName" resource extracted from the resource section of the PE.	STRING	• FILE • PROCESS • SECTION
REGVAL_DATA	Matches against registry value data in the context of a registry value set operation, either when a registry variable is created or its value is changed.	This data type is variable. You must specify it using the **-type** flag. Valid data types are the same as accepted by the Windows registry: • INT32	REGISTRY

Match type value	Description	Data type	Valid in object types
	You can use this *MATCH_type* value to control or filter the data being written or changed in a registry value.	• INT64 • BINARY • STRING • MULTI_STRING • EXPANDABLE_STRING	
REMOTE_MACHINE_ADDRESS	**Note:** This type is for reporting only. If used for matching, matches the specified type against file I/O initiated by a specific SMB client IP address in either IPv4 or IPv6 format. In other words, this type does not match for file I/O initiated on the local system going to an SMB server. It only matches for client I/O going to the local SMB server. This match type is mostly useful for generating event details.	STRING	This match type is valid in PROCESS **Initiator** (requires OBJECT_NAME to match SYSTEM:REMOTE) or FILE **Target** match.
SESSION_ID	Compares the specified match criteria against the session ID that the process/thread belongs to and can apply to both **Initiator** and **Target** objects.	UINT32	• PROCESS • THREAD
SHA1	Compares the SHA-1 hash sum of the	UINT8	• FILE • PROCESS

Match type value	Description	Data type	Valid in object types
	backing file with the match data. If the object is of type PROCESS or THREAD, the hash sum is calculated against its main executable module.		• SECTION • THREAD
SHA2_256	Compares the SHA2-256 hash sum of the backing file with the match data. If the object is of type PROCESS or THREAD, the hash sum is calculated against its main executable module.	UINT8	• FILE • PROCESS • SECTION • THREAD
SHA2_384	Compares the SHA2-384 hash sum of the backing file with the match data. If the object is of type PROCESS or THREAD, the hash sum is calculated against its main executable module.	UINT8	• FILE • PROCESS • SECTION • THREAD
SHA2_512	Compares the SHA2-512 hash sum of the backing file with the match data. If the object is of type PROCESS or THREAD, the hash sum is calculated against	UINT8	• FILE • PROCESS • SECTION • THREAD

Match type value	Description	Data type	Valid in object types
	its main executable module.		
STORAGE_BUS_TYPE	Compares the match criteria with the storage bus type that the disk is attached to.	UINT32	DISK
TARGET_OBJECT _NAME	Specifies the object name. Any combination of wildcards is accepted. Names follow the same conventions as OBJECT_NAME. But, they only match against the target of a file rename operation. This enables rules to be written that only apply to rename operations based on both source (OBJECT_NAME) and target (TARGET_OBJECT_NAME) name. • OBJECT_NAME is not required. If it is not specified, any source matches. • ACCESS_MASK for a rename is DELETE, because it's from the perspective of the source file, even if the OBJECT_NAME is not specified.	STRING	FILE

Match type value	Description	Data type	Valid in object types
USER_NAME	Matches the text representation of the user name.	STRING	• PROCESS • THREAD
USER_SID	Matches the text representation of the user account SID (that is, S-1-5-21-22-23-24-1168) .	STRING	• PROCESS • THREAD
VERSION_RESOURCE	Matches the “FileVersion” resource extracted from the resource section for the PE.	STRING	• FILE • PROCESS • SECTION
VERSION	Matches the version extracted from the resource section for the file.	STRING	• PROCESS • SECTION • THREAD
VTP_PRIVILEGES	Matches the bitmask against the VTP privileges of the target. The defined bits are: • PRIVILEGE_IOCTL (0x1) — Signed by a VTP-trusted certificate. • PRIVILEGE_ISG (0x8) — Signed by a **Trellix** certificate specifically. Files signed by Microsoft: • **VTP_TRUST** — Yes • **VTP_PRIVILEGES** — Yes	UINT64 - BITMASK	• FILE • PROCESS • THREAD

Match type value	Description	Data type	Valid in object types
	• **=0x08** — No • **=0x09** — Yes Files signed by **Trellix**: • **VTP_TRUST** — Yes • **VTP_PRIVILEGES** — Yes • **=0x08** — Yes • **=0x09** — Yes Files signed by 3rd party: • **VTP_TRUST** — No • **VTP_PRIVILEGES** — No • **=0x08** — No • **=0x09** — No		
VTP_TRUST	Checks if VTP trusts the process or file. The value is treated as Boolean. That is, a value of 1 in the match type matches only processes trusted by VTP. A value of 0 matches non-trusted processes.	UINT8	• PROCESS • SECTION • THREAD
WOW64	Matches a data value of "1" if the process/thread is a WOW64 process. This can only be true on 64-bit platforms and always matches a "0" on 32-bit platforms. This match can apply to both **Initiator** and **Target** objects.	UINT8	• PROCESS • THREAD

OBJECT_NAME guidelines

Use these guidelines when specifying the OBJECT_NAME match value in a *Match_type* value. You can use any combination of wildcards.

<table>
<tr><th>OBJECT_NAME value</th><th>Notes</th></tr>
<tr><td>Disk name</td><td>Accepted formats are:
HardDiskXX — HardDisk0
$(SystemDrive) — The disk that contains the system volume.</td></tr>
<tr><td>Fully qualified file path</td><td>Note: AAC doesn't support short paths.
System — Specifies the system process name. To match based on the thread running in the system process context, the rule must set an Initiator command to "System".
System:Remote — Specifies the system process name for remote systems. To match file operations for a remote system, the rule must set an Initiator command to "System:Remote".
To match based on both “System” and “System:Remote”, configure the rule to specify 2 matches or specify “System*”.</td></tr>
<tr><td>Fully qualified registry key/value path</td><td>These root keys are recognized:<table><tr><th>Key</th><th>Matches</th></tr><tr><td>HKLM</td><td>HKLM is equivalent to HKEY_LOCAL_MACHINE.</td></tr><tr><td>HKCU</td><td>All user registry keys (not just the current user) and</td></tr></table></td></tr>
</table>

OBJECT_NAME value	Notes	
		the .default user key. HKCU is equivalent to: • HKEY_CURRENT_USER • HKEY_USERS **Note:** Matching against specific user SIDs is not supported.
	HKCUC	All user classes (HKCU/*_CLASSES).
	HKCR	System classes and all user classes (HKCU/*_CLASSES). HKCR is equivalent to HKEY_CLASSES_ROOT.
	HKCCS	• HKLM/SYSTEM/CurrentControlSet • HKLM/SYSTEM/ControlSet00X
	HKLMS	• HKLM/Software on 32-bit and 64-bit systems • HKLM/Software/Wow6432Node

<table>
<tr><th>OBJECT_NAME value</th><th>Notes</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>on 64-bit systems only</td></tr>
<tr><td>HKCUS</td><td>• HKCU/Software on 32-bit and 64-bit systems
• HKCU/Software/Wow6432Node on 64-bit systems only</td></tr>
<tr><td>HKULM</td><td>• HKLM
• HKCU</td></tr>
<tr><td>HKULMS</td><td>• HKLMS
• HKCUS</td></tr>
<tr><td>HKALL</td><td>• HKLM
• HKU</td></tr>
</table>

Note: If the rule specifies a name where the root starts or contains a wild character, the AAC code performs no name normalization and that name might never match correctly. For example, **\mcshield\start is a valid name, but H*L*\mcshield\start is not.

HKEY_CURRENT_CONFIG is not supported.</td></tr>
<tr><td>Fully qualified section name</td><td></td></tr>
<tr><td>Process name or fully qualified process path</td><td>Process name must also be specified for thread objects.</td></tr>
<tr><td>Volume name</td><td>• Must be specified in the format:</td></tr>
</table>

OBJECT_NAME value	Notes
	Volume{35FC9B67-54AC-49ff-AB99-33FFA2999670} • $(SystemDrive) — Immutable and always applies to the system volume.

ACCESS_MASK flags

Use these flags with the ACCESS_MATCH *Match_type* value.

Flag	Applies to object types	Applies when
CONNECT_NAMED_PIPE	FILE (representing a named pipe)	Attempt to connect to a named pipe.
CREATE	• FILE • KEY • PROCESS • THREAD • SECTION	• File, Key, Process, or Thread is created. If the Target to be blocked is a process, specify the object type as SECTION rather than PROCESS. • File is open for execute (SECTION object). This doesn't mean that the SECTION object itself is created, rather that a SECTION object can be created. The SECTION object might not be created for execute.
DELETE	• FILE • KEY • PROCESS • THREAD	• File or Key (not registry values) is deleted or set security is called. • Process is opened with PROCESS_TERMINATE. • Thread is opened with THREAD_TERMINATE.

Flag	Applies to object types	Applies when
ENUM	• KEY • VALUE	• Key is opened with KEY_ENUMERATE_SUB_KEYS. • Values are enumerated with RegEnumValue.
EXECUTE	FILE	• File is opened with FILE_EXECUTE access. • SECTION object is created with SECTION_MAP_EXECUTE. **Tip: Best practice** Blocking SECTION objects might cause Windows to call a NtRaiseHardError(). To block loading unwanted code without this side-effect, use CREATE with SECTION. • Directory is opened with traverse access.
LOAD_IMAGE	SECTION	Notification only (cannot block the image load).
LOAD_KEY	KEY	Registry hive is loaded into a key with ZwLoadKey or RegLoadKey.
LOCK_RANGE		Attempt to lock or unlock a byte-range lock on a file. Use this access mask to protect a log file. You don't need to use this access mask for files that you aren't going to WRITE to at runtime, but byte-range locks don't stop reading and executing files.

Flag	Applies to object types	Applies when
OPEN_FOR_DELETE	FILE	Create/open event that requested delete access.
POST	FILE	Post-operation event. Events that carry this bit only match against rules that have this bit set. Also, if the access mask contains other bits set (not including POST), the rule evaluates to true only if at least one other bit matches the event.
QUERY	• KEY • VALUE	Attempt to query a registry key/value occurs.
READ	• FILE • KEY • VALUE	Existing file/key is being opened for read access. **Note:** This does not match with registry key/value enum/query operations. See ENUM and QUERY for matching against registry query/enum operations.
READ_DATA	FILE	An actual read file I/O occurs (ReadFile executed from user-space).
RENAME	• FILE • KEY • VALUE	Registry key or file rename operation occurs.
REPLACE_KEY	KEY	Registry key is replaced (RegReplaceKey).

Flag	Applies to object types	Applies when
RESTORE_KEY	KEY	Registry key is restored (RegRestoreKey).
SET_FILE_LENGTH	FILE	Any operation that changes the file length (ZwSetInformationFile), where class is one of: • FileEndOfFileInformation • FileAllocationInformation • FileValidDataLengthInformation This access bit helps with file-copy detection, when the destination file is extended and then written to.
SET_REPARSE	FILE	Attempt to set the reparse data on a file or directory object. Do not use this access mask with IS_DIRECTORY. Attempts to set a reparse point on an alternate data stream don't match correctly. This is because the file system always considers alternate data streams as "file" objects, even if the base file object is a directory. But, reparse data is configurable from an alternate data stream file handle on a directory, which causes STATUS_REPARSE to be returned for all streams of a directory or file object.
TERMINATING	• PROCESS • THREAD	Notification only (cannot block a terminate action).
WRITE	• FILE • KEY • VALUE	• Existing file is opened for write (FILE_GENERIC_WRITE

Flag	Applies to object types	Applies when
	• PROCESS	and disposition TRUNCATE_EXISTING). File rules, using this flag, and specifying the file name as a fully qualified path including drive letter, also matches rename operations for any of the upper-level directories. For example, if the rule specifies "c:\program files\mcafee\systemcore**", this rule matches rename operations against: ▫ c:\program files\mcafee\systemcore ▫ c:\program files\mcafee ▫ c:\program files\ But the rule doesn't match: ▫ c:\program files\microsoft ▫ c:\program files\mcafee\VSE • Existing key is opened for write (KEY_WRITE). • Process is opened for write access: ▫ PROCESS_CREATE_PROCESS ▫ PROCESS_CREATE_THREAD ▫ PROCESS_DUP_HANDLE ▫ PROCESS_SET_QUOTA ▫ PROCESS_SET_INFORMATION ▫ PROCESS_SUSPEND_RESUME ▫ PROCESS_VM_OPERATIONS ▫ PROCESS_VM_WRITE • Handle to the thread is opened with write access: ▫ THREAD_DIRECT_IMPERSONATION ▫ THREAD_IMPERSONATE ▫ THREAD_SET_CONTEXT ▫ THREAD_SET_INFORMATION

Flag	Applies to object types	Applies when
		▫ THREAD_SET_LIMITED_INFORMATION ▫ THREAD_SET_THREAD_TOKEN ▫ THREAD_SUSPEND_RESUME • Registry value is created, written, or deleted. Values are considered the data of a key.
WRITE_ATTRIBUTE	FILE	File or directory's attributes are written to.
WRITE_DATA	FILE	Actual write file I/O (WriteFile executing from user-space).

Commands to query system state

iDump command

The **iDump** command dumps global variables defined in the rule to the log file if debug logging is enabled.

Syntax

```
iDump filter
```

If *filter* is not specified, this command dumps all variables.

Parameter

Parameter	Description
filter	String that represents the names of the global variables to dump. The *filter* parameter can contain wildcards.

For more Expert Rules examples, visit the Trellix Github repository.

iEnv command

The **iEnv** command returns the specified environment variable value or an empty string if the variable does not exist.

Syntax

```
iEnv name
```

Parameter

Parameter	Returns
name	Value of the specified environment variable.

Example

```
set PingExe [iEnv SystemRoot]\\system32\\ping.exe
```

For more Expert Rules examples, visit the Trellix Github repository.

iList command

The **iList** command sorts the values in the list in ascending order and removes duplicate values.

Syntax

```
iList -d list
```

Parameter

Parameter	Description
-d	Indicates that the *list* contains directory names and converts all directory characters to the proper format for the operating system. Any duplicate separators are combined into one before the comparisons are done, any trailing directory characters are removed before the list is returned. If an entry is a subdirectory of another element, only the parent directory is returned.

Example

```
set alist {{c:/tmp\\ } {c:\tmp\a} {c:\tmp/b/c} {d:\debug}}
set blist [iList -d $alist]
```

The "blist" list now contains:

```
{{c:\tmp} {d:\debug}}
```

For more Expert Rules examples, visit the Trellix Github repository.

iReg command

The **iReg** command reads information from the local registry.

Syntax

```
iReg [-32] param
```

Parameters

To read the 32-bit hive on a 64-bit operating system, specify **-32** as the first argument.

Parameter	Description
open *keyname*	Opens a registry key named *keyname* and returns "1" if successful or "0" otherwise. Closes the key when the scanning session is over.
exist *keyname*	Tests to see if a registry key named *keyname* exists and returns "1" if it exists or "0" otherwise.
value *keyname valuename*	Reads information from the registry key *keyname* with the value name of *valuename*. If the value is type: • **string** — Returns the string value. • **int** — Returns the string value. • **MULTI_SZ** — Returns a Tcl list.
keys *keyname*	Returns a list of subkeys that exist under the key specified by *keyname*.

Parameter	Description
v_exists *keyname valuename*	Tests to see if the *valuename* item exists under the key specified by *keyname* and returns "1" if exists or "0" otherwise.

You can use the following shortcuts for the registry *keyname*.

Keyname	Shortcut
HKEY_LOCAL_MACHINE	HKLM
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_CONFIG	HKCC
HKEY_CURRENT_USER	HKCU
HKEY_USERS	HKUS

For example, to specify the software hive on the local system, use HKLM\\Software.

For more Expert Rules examples, visit the Trellix Github repository.

iSystem command

The iSystem command returns information about the client system where the rule is executed.

Syntax

```
iSystem param
```

Parameters

Parameter	Returns
version	Version of the operating system in the format ***major.minor.build***.

Parameter	Returns
major	Major version of the operating system.
minor	Minor version of the operating system.
build	Build number of the operating system.
csd	CSD value. Usually, this is the Service Pack in the form of a string, such as "Service Pack 1".
platform	String with the platform name, for example, "Windows 7".
type	System type: • Workstation • Server • Unknown
cpu_arch	CPU architecture: • 320 for 32-bit CPU • 640 for 64-bit AMD type CPU • 641 for 4-bit Itanium type CPU
os_arch	Operating system architecture: • 320 for 32-bit operating system • 640 for 64-bit operating system
install_dir	Location of the Windows installation directory.
sys32_dir	Location of the System32 folder.
users_folders ***folder_types***	List of folder locations for all users created on the system. You can specify the types of folders to return. The valid folder types are listed in `HKEY_USERS\<user sid>\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders`

Parameter	Returns
	In addition, you can specify these special folder types: • Temp — All temp folders on the system • Profile — All users' profile root folder • Downloads — All users' download locations • -no_defaults —All folders that are not changed from their default values.

For more Expert Rules examples, visit the Trellix Github repository.

iTerminate command

The **iTerminate** command stops building the rules and adds the specified message text to the error log.

Syntax

```
iTerminate "msg"
```

Parameter

Parameter	Description
msg	The message to add to the error log.

iUser command

The **iUser** command returns information about users on a system.

Syntax

```
iUser param
```

Parameters

Parameter	Returns
username	"1" if the user exists on the system, otherwise "0".

Parameter	Returns
list	List of all users on the system.
groups *username*	List of the groups a user belongs to.

For more Expert Rules examples, visit the Trellix Github repository.

iUtil command

The **iUtil** command converts the specified string into arguments and pass it to the function.

Syntax

```
iUtil cvt2args name
```

Parameter

Parameter	Description
cvt2args	Converts the specified strings into arguments.

Example

```
set test_var10 [iUtil cvt2args $test_var10]
```

For more Expert Rules examples, visit the Trellix Github repository.

Learn Expert Rules for Buffer overflow, Illegal API use and Services

Legacy McAfee Host IPS rule structure

Rules contain both required and optional sections, one section per line. Each section defines a rule category and its value. One section always identifies the class of the rule, which defines the rule's overall behavior. Optional sections vary according to the class of the rule.

Here is the basic structure of a **McAfee Host IPS** rule:

```
Rule {
        SectionA value
        SectionB value
        SectionC value
```

```
        ...
}
```

Because the structure and class types for legacy Expert Rules are identical to those in **McAfee Host IPS**, you can copy existing **McAfee Host IPS** rules into **Trellix ENS** Expert Rules.

Note

Trellix ENS doesn't support signatures with multiple rules.

Legacy Syntax

Wildcards

You can use wildcards for section values in Expert Rules.

Wildcard character	Represents
? (question mark)	A single character.
***** (one asterisk)	Multiple characters, including **/** and ****. **Note:** For paths and addresses, use ****** (2 asterisks) to include **/** and ****. Use ***** (one asterisk) to exclude **/** and ****.
& (ampersand)	Multiple characters except **/** and ****. Use **&** to match the root-level contents of a folder, but no subfolders. For example: `Include "C:\test\\&.txt"`
! (exclamation point)	Wildcard escape. For example: `Include "C:\test\\yahoo!.txt"`

Environment variables

Use environment variables to specify file and directory path names.

The **iEnv** command takes one parameter (the variable name) in square brackets **[ ]**.

Environment variable	Represents
iEnv SystemRoot	C:\winnt\, where C is the drive that contains the Windows System folder. For example: `Include [iEnv SystemRoot]\\system32\ \abc.txt`
iEnv SystemDrive	C:\, where C is the drive that contains the Windows System folder. For example: `Include [iEnv SystemDrive]\\system32\ \abc.txt`

Using the Include and Exclude keywords

When you select a section value as **Include**, the section works on the value indicated. When you select a section value as **Exclude**, the section works on all values except the one indicated.

The keywords **Include** and **Exclude** are supported for all sections except **directives** and **attributes**.

Enclose the **Include** and **Exclude** keywords in brackets { ... }.

Note

For a standard subrule, use a single backslash in file paths. The standard subrule translates the single slashes to required double slashes. For a subrule in an Expert Rule, use double backslashes in file paths. The expert subrule performs no translation.

For example, to monitor all text files in C:\test\:

```
files { Include C:\\test\\*.txt }
```

To monitor all files except the text files in C:\test\:

```
files { Exclude C:\\test\\*.txt }
```

Combine keywords to exclude values from a set of included values.

For example, to monitor all text files in folder C:\test\ except file abc.txt:

```
files { Include C:\\test\\*.txt }
files { Exclude C:\\test\\abc.txt }
```

Each time you add the same section with the same keyword, you add an operation.

For example, to monitor any text file in folder C:\test\ whose name starts with the string “abc”:

```
files { Include C:\\test\\*.txt }
files { Include C:\\test\\abc* }
```

Exclude takes precedence over **Include**. For example:

- If a single subrule includes a particular user marketing\jjohns and excludes the same user marketing\jjohns, the signature doesn't trigger even when the user marketing\jjohns performs an action that triggers the signature.
- If a subrule includes all users but excludes the particular user marketing\jjohns, the signature triggers if the user isn't marketing\jjohns.
- If a subrule includes user marketing* but excludes marketing\jjohns, the signature triggers only when the user is marketing\anyone, unless the user is marketing\jjohns, in which case it doesn't trigger.

Sections that are common to all class types

Use these sections when defining rules of all class types.

All section names are case sensitive. Section values are not case sensitive.

For sections that apply to a specific class type only, see the section lists for that class type.

Section	Value	Description	Required?
user_name	{Include/Exclude user's name or system account}	Specifies the users that rule applies to. Specify particular users or all users. • **Local users**: *machine name/local user name*	Yes

Section	Value	Description	Required?
		• **Domain users**: *domain name/domain user name* • **Local system**: Local/System Some remotely initiated actions don't report the ID of the remote user, but use the local service and its user context instead. You must plan accordingly when developing rules. When a process occurs in the context of a Null Session, the user and domain are "Anonymous". If a rule applies to all users, use the * wildcard.	
Executable	{Include/Exclude file path name, fingerprint, signer, or description}	Specifies the executables that the rule applies to. Specify each executable inside brackets using: • -path — File path name • -hash — MD5 hash • -sdn — Signer • -desc — Description Each section can have multiple brackets and, inside the brackets, one or more options. The **-path**, **-sdn**, and **-desc** values are strings and must be Tcl-escaped if they contain	Yes

Section	Value	Description	Required?
		spaces or other Tcl-reserved characters. The **-hash** value is a 32-character hexbin string. For example: `Executable { Include -path "C:\\Program Files (x86)\ \McAfee Endpoint Security\\ Threat Prevention\ \mfetp.exe" -sdn "CN=\"McAfee, Inc.\", OU=Engineering, O=\"McAfee, Inc.\", L=Santa Clara, ST=California, C=US" -desc "on-access scanner service" }` If a rule applies to all executables, use the ***** wildcard.	
directives	operation type	Specifies the class-dependent operation types. For the operation type, see the directives in each class type description.	Yes
dependencies	{Include/Exclude “ID of a rule”}	Defines dependencies between rules and prevents triggering dependent rules. Add the dependencies section to prevent a more general rule from being triggering with a more specific rule. For example, use ID	No

Section	Value	Description	Required?
		428 for Buffer Overflow signatures.	
attributes	-no_log	**Sends no events from the signature to the Trellix ePO - On-prem server. Sends no events from the signature.**	No
	-not_auditable	Generates no exceptions for the signature when Adaptive mode is enabled.	
	-no_trusted_apps	Specifies that the trusted application list doesn't apply to this signature.	
	-inactive	Disables the signature.	

Class types

Buffer Overflow class type

The **Buffer Overflow** class type prevents buffer overflow exploits for applications in the application protection list.

Section	Value	Notes
user_name		
Executable		

Section	Value	Notes
dependencies	428	Specifies Signature 428, Generic Buffer Overflow, a generic buffer overflow rule. (Optional) We recommend including section "dependencies 428" to avoid triggering the generic signature.
caller module	Path to a module (for example, a DLL) loaded by an executable that calls and causes a buffer overflow	
directives	bo:stack	Examines the memory location that is executing and detects if it is running from writable memory that is part of the current thread's stack.
	bo:heap	Examines the memory location that is executing and detects if it is running from writable memory that is part of a heap.
	bo:writeable_memory	Examines the memory location that is executing and detects if it is running from writable memory that is not part of the current thread's stack or a heap.
	bo:invalid_call	Checks that an API is called from a proper call instruction.
	bo:target_bytes	A hexadecimal string representing 32 bytes of instructions that can be used to create a targeted exception for a false positive without disabling

Section	Value	Notes
		buffer overflow for the entire process.
	bo:call_not_found	Checks that the code sequence before the return address isn't a call.
	bo:call_return_unreadable	Checks that the return address isn't readable memory.
	bo:call_different_target_address	Checks that the call target doesn't match the hooked target.
	bo:call_return_to_api	Checks that the return address is an API entry point.

Illegal API Use class type

The **Illegal API Use** class type prevents illegal use of the Exploit Prevention API.

Section	Value	Notes
user_name		
Executable		
vulnerability_name	Name of the vulnerability	
detailed_event_info	One or more CLSIDs.	This value is a 128-bit number that represents a unique ID for a software component, such as: `"{FAC7A6FB-0127-4F06-9892-8D2FC56E3F76}"`
directives	illegal_api_use:bad_parameter	
	illegal_api_use:invalid_call	

Use this class to create a custom killbit signature. The killbit is a security feature in web browsers and other applications that use ActiveX. A killbit specifies the object class identifier (CLSID) for ActiveX software controls that are identified as security vulnerability threats. Applications that use ActiveX don't load specified ActiveX software with a corresponding killbit in place.

The primary purpose of a killbit is to close security holes. Killbit updates are typically deployed to Microsoft Windows operating systems using Windows security updates.

Here is an example of a killbit signature:

```
Rule {
tag "Sample4"
Class Illegal_API_Use
Id 4001
level 4
Executable { Include "*"}
user_name { Include "*"}
vulnerability_name {Include "Vulnerable ActiveX Control Loading ?"}
detailed_event_info { Include
"0002E533-0000-0000-C000-000000000046"\"0002E511-0000-0000-C000-000000000046"}
directives files:illegal_api_use:bad_parameter illegal_api_use:invalid_call
attributes -not_auditable
}
```

Services class type

The `Services` class type protects Windows Services operations.

Section	Values	Notes
user_name		
Executable		
services	Name of the service to protect.	(Required) The name of the service is in the corresponding registry key under HKLM_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\.
display_names	Display name of the service.	Required. This name appears in the Services manager and in the registry value HKLM_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<name-of-service>\

Section	Values	Notes
directives	services:delete	Deletes a service.
	services:create	Creates a service.
	services:start	Starts a service.
	services:stop	Stops a service.
	services:pause	Pauses a service.
	services:continue	Continues a service after a pause.
	services:startup	Changes the startup mode of a service.
	services:profile_enable	Enables a hardware profile.
	services:profile_disable	Disables a hardware profile.
	services:logon	Changes the logon information of a service.

Troubleshooting Expert rules

This example log shows some of the possible cause of errors while writing Expert Rules. **Trellix Endpoint Security (ENS)** provides information in the EndpointSecurityPlatform_Errors.log file about rules that didn't successfully compile and so were not enforced.

Because all Expert Rules are compiled into a single group, when an Expert Rule generates an error, no Expert Rules are enforced.

Tip

Best practice: To isolate any potential issues, every time you create a rule, verify that it was successfully enforced on the client system.

The EndpointSecurityPlatform_Errors.log file includes detailed information, such as the content of the rule and the parameter that caused the error. For example, this log error shows the Expert Rules error, which is an extra `Include` command:

```
08/11/2017 11:57:34.403 AM   mfeesp(4016.4412) <SYSTEM> ApBl.AP.Error: Syntax error: Include: Invalid
number of arguments
    while executing
"Include Include OBJECT_NAME  { -v "*PowerShell*" }"
        Include Include OBJECT_NAME  { -v "*PowerShell*" }
        Include PROCESS_CMD_LINE { -v "*-extoff* script.scp" }
        Include ..."
    invoked from within
"Process {
        Include OBJECT_NAME      { -v "*PowerShell*" }
        Include PROCESS_CMD_LINE { -v "*-extoff*" }
        Include PROCE ..."
    invoked from within
"Rule -id "4100" {
    Reaction BLOCK
    Group "ExPExpertRules"
    Description "testrule"
    Process {
         Include AggregateMatch {
         Include OBJECT_NAME      {  ..."
    invoked from within
"Policy {
Rule -id "4100" {
    Reaction BLOCK
    Group "ExPExpertRules"
    Description "testrule"
    Process {
         Include AggregateMatch {
         Include OBJECT_NA ..."LastErr 0x000010dd The operation identifier is not valid.
08/11/2017 11:57:34.403 AM   mfeesp(4016.4412) <SYSTEM> ApBl.AP.Error: ERR: BLError 0xc0380102, Could not
process content file
```

COPYRIGHT

Trellix