ESET Inspect On-Prem

Rules guide

Click here to display the online version of this document

Digital Security
Progress. Protected.

1 Rules guide 1
2 Rule syntax 1
3 Operations 6
4 Actions 15
5 Targets 17
6 Extended incident rules 18
6.1 ApiCall 22
6.2 ClientEnterprise 23
6.3 ClientFileItem 23
6.4 ClientLiveGrid 23
6.5 ClientModule 24
6.6 ClientProcessInfo 24
6.7 CodeInjectionInfo 24
6.8 DateTime 25
6.9 DnsInfo 26
6.10 Endpoint 27
6.11 Enterprise 28
6.12 EnterpriseInspector 28
6.13 FileAttribute 29
6.14 FileItem/DestFileItem 29
6.15 InspectDetection 31
6.16 LiveGrid 31
6.17 Module 32
6.18 Network 34
6.19 OpenProcess 36
6.20 ProcessBehavior 37
6.21 ProcessInfo 37
6.22 RegistryItem 38
6.23 Scripts 39
6.24 SystemInfo 40
6.25 TargetUser/DoneByUser 40
6.26 UserGroupData 41
6.27 UserLogonData 42
6.28 WmiExecutionInfo 42
6.29 WmiPersistenceInfo 43
6.30 WmiQueryInfo 44
7 Property Types & Relations, Symbols 45
8 Supported environment variables 48
9 Best Practices 50
10 Rules Examples 51
10.1 Working with registry 52
10.2 Monitoring network connections 54
10.3 Working with URLs 57
10.4 Working with command line 59
10.5 Working with a parent-child relationship 61
10.6 Working with LiveGrid and Safe property 63
10.7 Working with compromised flag 66

Rules guide

A rule is defined using XML-based language.

Rules are evaluated on the Endpoint by default. A matched rule triggers associated actions and notifies a security engineer by raising a detection. The detection is displayed in the Detections view. It is also exported to ESET PROTECT (or SIEM), or an email can be automatically sent when the detection is triggered.

Rule syntax

A rule is a set of expressions defined using XML-based language.

ESET Inspect Web Console contains a pre-defined set of rules, but you can add and edit your own rules.

The general body structure of a rule is:

```
<rule>
 <definition>
         <ancestor></ancestor>
         <parentprocess></parentprocess>
         <process></process>
         <operations>
                 <operation></operation>
         </operations>
 </definition>
 <description>
         <name></name>
         <category></category>
         <explanation></explanation>
                 <os></os>
         <mitreattackid></mitreattackid>
         <maliciousCauses></maliciousCauses>
         <benignCauses></benignCauses>
         <recommendedActions></recommendedActions>
 </description>
 <maliciousTarget name=""/>
 <actions>

         <action name="action" />

 </actions>

</rule>
```

Definition Tag

ancestor—can take an additional attribute:

- **distance**—specifies the actual distance from the current process of the ancestor being matched, i.e., 1 is the parent process, 2 grandparent, ... If unspecified, the property is matched against all process ancestors
- **unique**—enables the ancestor tag to remove duplicate processes in the ancestor tree. For example, the malware creates multiple instances of the same process to avoid detection (explorer -> cmd -> cmd -> cmd -> malware), and this attribute will get rid of these duplications (explorer -> cmd -> malware)

Process and parentprocess

The `process` part lets security engineers restrict events to a specific process; therefore, you can write rules like "Outlook creates EXE file". If the process element is empty, `parentprocess` and `operations` are evaluated for all the processes.

The `parentprocess` part is similar to the process but allows security engineers to test parent process attributes. This enables rules like "PowerShell, started by Word, connects to the internet".

`process`, `parentprocess`, `ancestor` and `operation` use an expression element to describe a logical expression which is evaluated to check if a detection should be triggered. An expression consists of conditions and logical operators. A condition checks the value of some property, and logical operators group these conditions into logical expressions.

Expression example:

```
<operator type="and">

    <condition component="FileItem" property="Path" condition="notstarts" value="%SY
STEM%" />

    <operator type="or">

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="svchost" />

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="smss" />

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="lsass" />

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="csrss" />
```

```
      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="wininit" />

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="services" />

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="winlogon" />

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="system" />

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="userinit" />

      <condition component="FileItem" property="FileNameWithoutExtension" condition=
"is" value="conhost" />

  </operator>

</operator>
```

Operations

The `operations` part defines which operations executed by a process raise the detection. If empty, the detection is triggered when the process generates an event.

Operations are defined using an operation element with a `type` attribute and an expression element.

```
<operation type="WriteFile">Expression</operation>
```

For the complete list of supported operations, see the Operations topic.

Operator

Supported logical operators are: AND, OR, and NOT. Logical operators can be nested. Therefore, a logical operator can be an argument of another logical operator. Apart from logical operators, a condition element can be used as an argument for logical operators. You can use the operator tag as shown in this example:

```
<process>

    <operator type ="AND">

        <condition component ="FileItem" property ="Path" condition ="starts" value
="% TEMP %"/>

        <condition component ="FileItem" property ="FileName" condition ="is" value
="svchost"/>

    </operator>

</process>
```

Condition elements consist of three parts:

- Property of operation's argument or process
- A value specified by the rule's author
- Relationship between the value and the property

Properties are hierarchically grouped into "components".

The **Severity** of rules defines the way a rule is displayed in the Detections screen. There are three levels of severity: 1–39 > **Info**, 40–69 > **Warning**, and 70–100 > **Threat**.

Property types and their relations (condition attributes) are:

	is(not)set	is(not)	is(not)empty	(not)starts	(not)contains	(not)ends	less, lessOrEqual, greater, greaterOrEqual
String	✓	✓	✓	✓	✓	✓	
Int	✓	✓			✓		✓
Value	✓	✓	✓				
Bool	✓	✓					
Date	✓	✓					✓
Set of Strings	✓		✓		✓		
IPv4 Address	✓	✓	✓				
IPv6 Address	✓	✓	✓				
Set of IPv4 Addresses	✓		✓		✓		
Set of IPv6 Addresses	✓		✓		✓		

Description Tag

description is mandatory and must contain a name and category, but the rest is optional. It is helpful to define the fields listed below because they appear in the Detections details of ESET Inspect

name—unique name of the rule. It is shown in the list of rules.

category—allows you to categorize rules. You can specify your own categories.

explanation—explains the reason why the rule is triggered

os—this tag contains the operating system to which the rule applies. Possible values are: Windows, Linux, OSX, ANY

mitreattackid—contains the id of the MITRE ATT&CK®

maliciousCauses—describes the malicious causes of the event or the change that triggered the rule

benignCauses—describes the benign causes of the event or the change that triggered the rule

recommendedActions—describes the recommended actions to be taken by security engineers. A user can use the following markdown:

- [navigation:computer_details]—the default name shown to the user will be **Computer Details**
- [navigation:executable_details]—the default name shown to the user will be **Executable Details**
- [navigation:process_details]—the default name shown to the user will be **Process Details**
- [remediation:shutdown]—the default name shown to the user will be **Shutdown Computer**
- [remediation:reboot]—the default name shown to the user will be **Reboot Computer**
- [remediation:kill]—the default name shown to the user will be **Kill Process**
- [remediation:block]—the default name shown to the user will be **Block Hash**
- [misc:download]—the default name shown to the user will be **Download file**

Additionally, each command can display alternative text instead of the default one. To specify alternative text to display, follow the command with pipe sign and text. For example, [navigation:computer_details|GoToComDet] will display **GoToComDet** instead of default **Computer Details**. For other examples of use, search for the rule c0601 in the **Detection rules** tab in the **Audit** main window.

guid—used for internal rules. External rules have guid automatically generated. It is used to uniquely identify rules during export/import.

maliciousTarget Tag

You can use the **maliciousTarget** tag to specify the target that will be affected by user actions. For example, when you select block executable, the changes depend on the maliciousTarget. If you specify maliciousTarget as **current** or **parent**, the block executable user action will change to blockProcessExecutable or blockParentProcessExecutable, respectively. It does not change the behavior of the Actions Tag. Possible values are **current**, **module**, **none** and **parent**.

The **maliciousProcess** tag has been replaced by the **maliciousTarget** tag. It is still in legacy support but, we recommend using the **maliciousTarget** tag.

Actions Tag

Actions tag allows you to specify a set of actions that are executed when the rule is triggered. For the complete list of supported actions, see the Actions topic.

You can put actions into a single action element:

```
<action name="BlockProcessExecutable"/>
```

Or stack multiple action elements into single actions element (as shown in the following example):

```
<actions>

<action name="BlockProcessExecutable"/>
```

```
…
</actions>
```

Multiple actions can be triggered from a single rule.

Operations

The `operations` part defines which operations executed by a process raise the detection. If empty, the detection is triggered when the process generates an event.

Operations are defined using an operation element with a `type` attribute and an expression element.

```
<operation type="WriteFile">Expression</operation>
```

The following components are supported by all operation.

- DateTime
- EnterpriseInspector
- SystemInfo

CodeInjection

A process was subject to some form of code injection.

Supported components:

- ClientProcessInfo
- CodeInjectionInfo
- Enterprise
- FileItem
- LiveGrid
- Module
- ProcessInfo

CreateNamedPipe

A named pipe was created.

Supported components:

- FileItem

CreateProcess

A process was created.

Supported components:

- Enterprise
- FileItem
- LiveGrid
- Module
- ProcessInfo

DeleteFile

A file was deleted.

Supported components:

- FileItem

Detection

This operation can be used in two different ways:

- Used in a regular rule—an event was triggered on client-side antivirus
- Used in a sequence rule—only detections triggered by Inspect

Supported components:

- Endpoint
- InspectDetection
- Network

DnsRequest

A DNS request was made (usually IP > domain, domain > IP).

Supported components:

- DnsInfo

HttpRequest

A HTTP request was made.

Supported components:

- Network

LoadDLL

A DLL was loaded.

Supported components:

- Enterprise
- FileItem
- LiveGrid
- Module
- ProcessInfo

LoadDriver

A driver or kernel module was loaded.

Supported components:

- FileItem

ModuleDrop

An executable was dropped.

Supported components:

- FileItem

MultipleFilesChanged

Process modified multiple files.

Supported components:

- ProcessBehavior

OpenProcess

Added a new rule attribute, which triggers when a process is opened. Only the open process to lsass.exe is monitored.

Supported components:

- FileItem
- OpenProcess

ReadFile

Triggered when a monitored file was read. Monitored files refer to those which contain either sensitive information or stored credentials, for example, stored browser passwords, stored FTP clients passwords, AD database and so on.

Supported components:

- FileItem

RegDeleteKey

A registry key was deleted.

Supported components:

- RegistryItem

RegDeleteValue

The value of the registry was deleted.

Supported components:

- RegistryItem

RegRenameKey

A registry key was renamed.

Supported components:

- RegistryItem

RegSetValue

A registry key was altered.

Supported components:

- RegistryItem

RenameFile

A file was renamed.

Supported components:

- FileItem
- DestFileItem

Scripts

A script exposed by AMSI was executed.

Supported components:

- Scripts

SetFileAttribute

A file attribute was set.

Supported components:

- FileAttribute
- FileItem

SystemApiCall

A system function was called.

Supported components:

- ApiCall

TcpIpAccept

An incoming TCP/IP connection was accepted.

Supported components:

- Network

TcpIpConnect

An outbound TCP/IP connection was made.

Supported components:

- Network

TcpIpProtocolIdentified

On top of TCP/IP connection, describes the protocol used.

Supported components:

- Network

TruncateFile

A file was truncated, this operation is reported only on Posix systems

Supported components:

- FileItem

UserActivate

The user was activated.

Supported components:

- TargetUser
- DoneByUser
- UserGroupData

UserAddToGroup

The user was added to the group.

Supported components:

- TargetUser
- DoneByUser
- UserGroupData

UserCreate

A new user was created.

Supported components:

- TargetUser
- DoneByUser
- UserGroupData

UserDisable

The user was disabled.

Supported components:

- TargetUser
- DoneByUser
- UserGroupData

UserLogin

The user logged in.

Supported components:

- TargetUser
- DoneByUser
- UserGroupData
- UserLogonData

UserLogout

The user logged out.

Supported components:

- TargetUser
- DoneByUser
- UserGroupData
- UserLogonData

UserRemove

The user was removed.

Supported components:

- TargetUser

- DoneByUser
- UserGroupData

UserRemoveFromGroup

The user was removed from the group.

Supported components:

- TargetUser
- DoneByUser
- UserGroupData

WmiExecution

WMI execution event was triggered.

Supported components:

- ClientEnterprise
- ClientFileItem
- ClientLiveGrid
- ClientModule
- ClientProcessInfo
- FileItem
- WmiExecutionInfo

WmiPersistence

The event is generated when consumer binds to a filter.

Supported components:

- WmiPersistenceInfo

WmiQuery

WMI query was executed on a computer.

Supported components:

- ClientEnterprise
- ClientFileItem
- ClientLiveGrid
- ClientModule
- ClientProcessInfo
- WmiQueryInfo

WriteFile

A file was written to.

Supported components:

- FileItem

Earlier versions of Windows do not produce WMI events. This functionality has been available since Windows 10 version 1803.
Some of the events provide only partial information:
- File write events—only the first file change is recorded (This is per process. If two processes change the same file, both changes are recorded)
- Registry related events—only the first registry key change is recorded (first time by a process)
- DLLLoad—only DLLs that are not whitelisted by AV are recorded
- TcpIp events—only the first connection is recorded (first time by a process)
- Http events—only the first request is recorded (first time by a process)
- ModuleDrop (a.k.a PEDrop)—it is reported only for the first drop of a given module (first time on a computer)
- AmsiTriggerEvent—only the first execution is recorded (first time on a computer)

Actions

Actions tag allows you to specify a set of actions that are executed when the rule is triggered. Action names are:

- **BlockModule**—blocks DLL that is being loaded in the LoadDll event
- **BlockParentProcessExecutable**—blocks a parent process hash (only if not trusted or LiveGrid® info is missing)

- **BlockProcessExecutable**—blocks a process hash (ban hash via the rule, only if not trusted or LiveGrid® info is missing)
- **BlockProcessSuspiciousModules**—blocks a module marked as suspicious by MarkModuleSuspicious action
- **CleanAndBlockModule**—blocks dropped module in the ModuleDrop event
- **CleanAndBlockParentProcessExecutable**—cleans and blocks a parent process hash (only if not trusted or LiveGrid® info is missing)
- **CleanAndBlockProcessExecutable**—cleans and blocks a process hash (only if not trusted or LiveGrid® info is missing)
- **CleanAndBlockProcessSuspiciousModules**—cleans and blocks a module marked as suspicious by MarkModuleSuspicious action
- **DropEvent**—drops an event that triggered the rule
- **HideCommandLine**—do not save the command line of the proccess that triggered the rule
- **IsolateFromNetwork**—isolates the computer from the network
- **KillParentProcess**—kills parent of the running process that triggered the detection (only if not trusted or LiveGrid® info is missing)
- **KillProcess**—kills running process that triggered the detection (only if not trusted or LiveGrid® info is missing)
- **LogOutUser**—logs out the user from the operating system
- **MarkAsCompromised**—the process that triggered the rule will be marked as compromised. This status is visible in the process details view in ESET Inspect Web Console.
- **MarkAsResolved**—marks the currently evaluated detection as resolved
- **MarkAsScript**—marks an executable as a script
- **MarkModuleSuspicious**—marks a module as suspicious
- **Reboot**—reboots computer that triggered the detection
- **ReportIncident**—creates incident when the detection is triggered. You can aggregate detections into one incident using aggregateOn parameter. To specify time aggregation you can use aggregationParameter

Possible aggregateOn parameter values are:

  - Computers
  - Time
  - TimeAndComputers

- **Shutdown**—shutdowns computer that triggered the detection
- **StoreEvent**—stores events that triggered the detection from this rule regardless of other settings. You can

use it if the events are not stored by default

• **SubmitModuleToLiveGuard**—submits module to ESET LiveGuard

• **SubmitParentToLiveGuard**—submits parent of the executable that triggered the detection to ESET LiveGuard

• **SubmitToLiveGuard**—submits executable that triggered the detection to ESET LiveGuard

• **TriggerDetection**—if you do not specify actions in the actions tag field, this action is executed by default and the detection triggers in ESET Inspect. If other actions are specified, and you still want to trigger detection, you must add this action

Some of the actions are disabled for Linux and macOS:

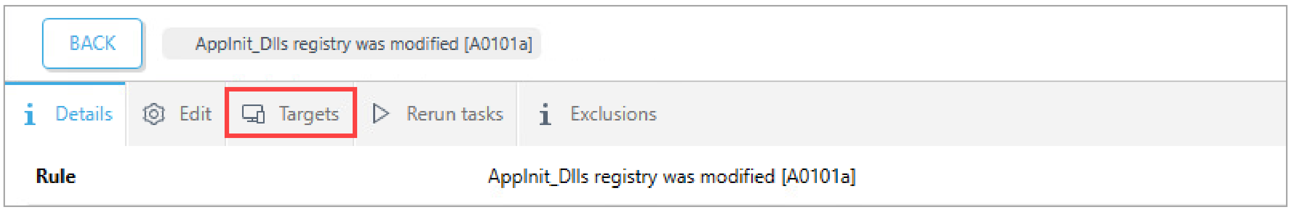
- IsolateFromNetwork
- KillProcess
- KillParentProcess
- SubmitModuleToLiveGuard
- SubmitParentToLiveGuard
- SubmitToLiveGuard

Targets

You can specify which computers or groups the rule should target.

1.Select a rule you want to change the targets for and click **Details**

2.Click **Targets** tab

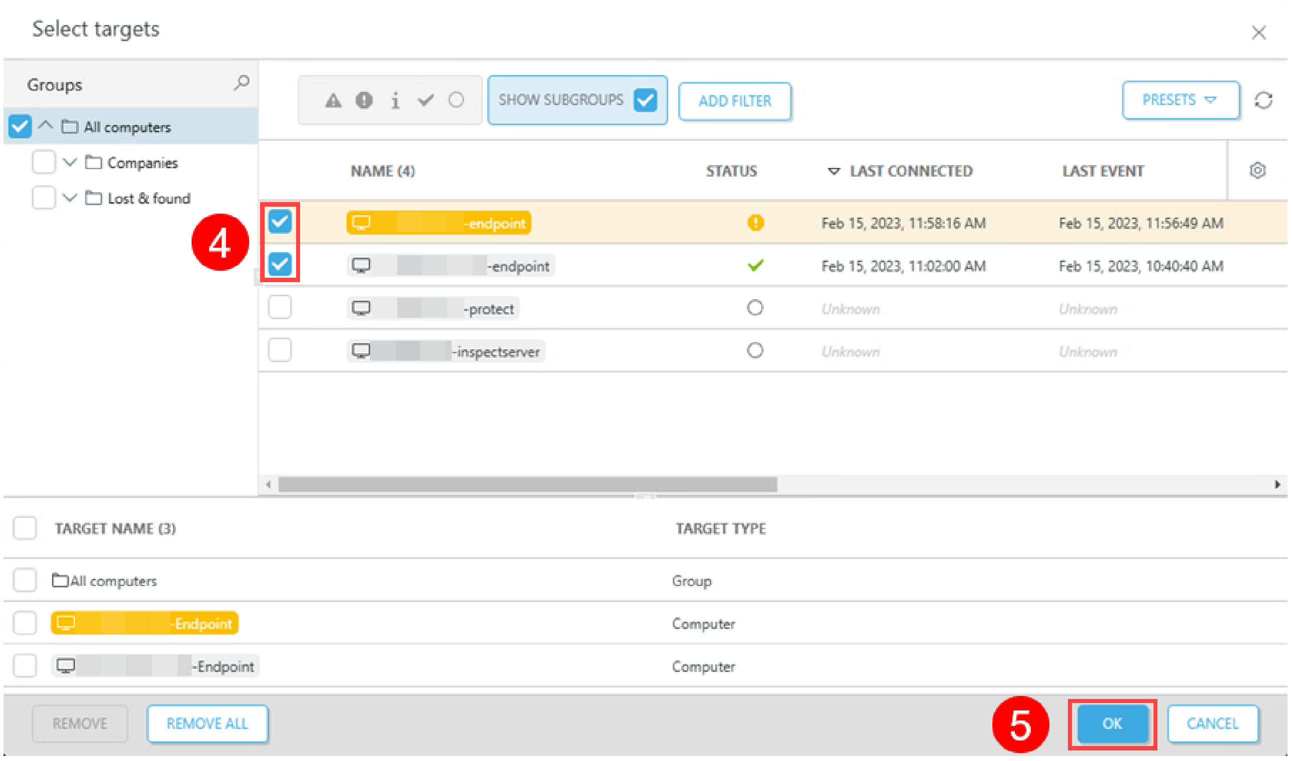


Rule AppInit_Dlls registry was modified [A0101a]

3.Click **Assign** button

4.Select the computers or groups you want the rule to be assigned to

5.Click **Ok**



Select targets

Groups
- ✓ All computers
  - Companies
  - Lost & found

SHOW SUBGROUPS ✓ | ADD FILTER | PRESETS

	NAME (4)	STATUS	LAST CONNECTED	LAST EVENT
✓	[redacted]-endpoint	!	Feb 15, 2023, 11:58:16 AM	Feb 15, 2023, 11:56:49 AM
✓	[redacted]-endpoint	✓	Feb 15, 2023, 11:02:00 AM	Feb 15, 2023, 10:40:40 AM
	[redacted]-protect	○	Unknown	Unknown
	[redacted]-inspectserver	○	Unknown	Unknown

TARGET NAME (3)	TARGET TYPE
All computers	Group
[redacted]-Endpoint	Computer
[redacted]-Endpoint	Computer

REMOVE | REMOVE ALL | OK | CANCEL

Extended incident rules

Threshold rules

Threshold rules enable you to create incidents when a defined number of detections occur in a given time period.

Example:

```
<?xml version="1.0" encoding="utf-8"?>

<rule>

  <definition>

    <threshold count="3" interval="900s">

      <detection>

        <definition>

          <process>

            <operator type="or">

              <condition component="FileItem" property="FileName" condition="is" val
ue="notepad.exe"/>

              <condition component="FileItem" property="FileName" condition="is" val
ue="cmd.exe"/>

            </operator>
          </process>
          <operations>
            <operation type="Detection">
              <operator type="or">
                <condition component="InspectDetection" property="RuleName" conditio
n="contains" value="PB000"/>
                <condition component="InspectDetection" property="RuleName" conditio
n="contains" value="PB001"/>
              </operator>
            </operation>
          </operations>
        </definition>
      </detection>
      <cardinality>
        <property name="computerName" value="1" />
        <property name="ruleName" value="2" />
      </cardinality>
    </threshold>
  </definition>
  <actions>
    <action name="ReportIncident" aggregateOn="TimeAndComputers" aggregationParamete
r="8h"/>
  </actions>
  <description>
    <name>threshold rule</name>
    <category>default</category>
  </description>
</rule>
```

Things to notice in the rule example above:

1.This rule is defined with a **threshold** which has two parameters:

a.count—the number of occurrences for the action to trigger.

b.interval—tied to the cardinality. For the count to increase, the cardinality condition has to be met within this time frame.

2.The first condition for the count to increase is a detection from a process where the filename is either **notepad.exe** or **cmd.exe**.

3.The second condition is **InspectDetection**, where the triggered rule name contains either **PB000** or **PB001**.

4.The third condition is **cardinality**. In the example above, for the count to increase, the detection must occur on one unique computer and in two unique rules.
Possible values for cardinality are:

a.computerName

b.ruleName

5.If all three conditions are met, the count is increased.

6.The only available action is **ReportIncident**.

Sequence rules

Sequence rules enable you to create incidents when detections occur in a specific sequence.

Example:

```
<?xml version="1.0" encoding="utf-8"?>

<rule>

  <definition>

    <sequence count="2" maxSpan="1m">

      <detection>

        <definition>

          <process>

            <condition component="FileItem" property="FileNameWithoutExtension" cond
ition="is" value="notepad" />

          </process>

          <operations>

            <operation type="Detection">

              <condition component="InspectDetection" property="RuleName" condition=
"contains" value="Rule 01"/>

            </operation>

          </operations>
        </definition>

      </detection>

      <detection>

        <definition>

          <process>

            <condition component="FileItem" property="FileNameWithoutExtension" cond
ition="is" value="notepad" />

          </process>

          <operations>

            <operation type="Detection">

              <operator type="and">

                <condition component="InspectDetection" property="RuleName" conditio
n="contains" value="Rule 02"/>

                <condition component="InspectDetection" property="RuleCategory" cond
ition="is" value="Custom category"/>

                <condition component="InspectDetection" property="RuleSeverity" cond
ition="is" value="Threat"/>

              </operator>

            </operation>

          </operations>

        </definition>

      </detection>

      <aggregateOn>

        <property name="Computer"/>

        <property name="Process"/>

        <property name="ParentProcess"/>

      </aggregateOn>

    </sequence>

  </definition>

  <description>

    <name>Notepad triggered sequence of detections</name>

    <category>
```

```
      Default
    </category>
  </description>
  <actions>
    <action name="ReportIncident"/>
  </actions>
</rule>
```

Things to notice in the rule example above:

1. The **sequence** tag specifies how many times the entire sequence has to match for the incident to be created and the maximum time between the first detection and the last detection in the sequence.

2. In the example above, for the sequence rule to trigger, **Rule 01** and **Rule 02** have to trigger a detection in that order twice in the span of 1 minute.

3. The **aggregateOn** tag specifies the conditions for which rule triggers should be grouped together.
   Possible values are:
   - Computer
   - Process
   - ParentProcess

4. The only available action is **ReportIncident**.

ApiCall

Returns information about API calls.

Property	Type	Description	Example
ApiName	String	Name of the API called by the process	Supported values are: • SetWinEventHook • RegisterRawInputDevices • SetWindowsHookEx • GetAsyncKeyState • CredEnumerate • CredReadDomainCredentials • CredFindBestCredential • CredRead • CredReadByTokenHandle • VaultEnumerateCredentials • RawSocketCreated (Linux only) • SocketFilterAttached (Linux only)

Example:

```
<operations>
    <operation type="SystemApiCall">
        <condition component="ApiCall" property="ApiName" condition="is" value="Regi
sterRawInputDevice" />
    </operation>
</operations>
```

Supported Operations and their components:

	ApiCall
SystemApiCall	X

ClientEnterprise

ClientEnterprise is available only in combination with the WmiExecution operation, which has a client process. A client process is a process that actually executed a WMI method. It shares the same properties as its counterparts without the 'Client' prefix.

Supported Operations and their components:

	ClientEntrprise
WmiExecution	X
WmiQuery	X

ClientFileItem

ClientFileItem is available only in combination with the WmiExecution operation, which has a client process. A client process is a process that actually executed a WMI method. It shares the same properties as its counterparts without the 'Client' prefix.

Supported Operations and their components:

	ClientEntrprise
WmiExecution	X
WmiQuery	X

ClientLiveGrid

ClientLiveGrid is available only in combination with the WmiExecution operation, which has a client process. A client process is a process that actually executed a WMI method. It shares the same properties as its counterparts without the 'Client' prefix.

Supported Operations and their components:

	ClientEntrprise
WmiExecution	X
WmiQuery	X

ClientModule

ClientModule is available only in combination with the WmiExecution operation, which has a client process. A client process is a process that actually executed a WMI method. It shares the same properties as its counterparts without the 'Client' prefix.

Supported Operations and their components:

	ClientEntrprise
WmiExecution	X
WmiQuery	X

ClientProcessInfo

ClientProcessInfo is available only in combination with the WmiExecution operation, which has a client process. A client process is a process that actually executed a WMI method. It shares the same properties as its counterparts without the 'Client' prefix.

Supported Operations and their components:

	ClientProcessInfo
WmiExecution	X
WmiQuery	X
CodeInjection	X

CodeInjectionInfo

For CodeInjection events, it reports information regarding the code injection technique used.

Property	Type	Description	Example
CodeInjectionType	Int	Code injection technique used	Possible values are: • CreateRemoteThread • SetThreadContext • ApcQueue

Supported Operations and their components:

	CodeInjectionInfo
CodeInjection	X

DateTime

Allows you to write rules that are triggered only at a specified time.

Property	Type	Description	Example
DayOfWeek	Int	Trigger the rule only on certain days of the week	Possible values are: • 0—Sunday • 1—Monday • 2—Tuesday • 3—Wednesday • 4—Thursday • 5—Friday • 6—Saturday
Hour	Int	Trigger the rule only in certain parts of the day. Possible values are 0 to 23	12
Minute	Int	Trigger the rule only in certain parts of the day. Possible values are 0 to 59	30

Example:

This rule is triggered every time the Notepad is opened from Monday to Friday between 9 a.m. to 5 p.m.

```
<definition>

 <process>

         <operator type="and">


 <condition component="FileItem" property="FileNameWithoutExtension" condition="is"
value="Notepad" />


 <condition component="DateTime" property="Hour" condition="greaterOrEqual" value="9
" />


 <condition component="DateTime" property="Hour" condition="lessOrEqual" value="17"
/>


 <condition component="DateTime" property="DayOfWeek" condition="greaterOrEqual" val
ue="1" />


 <condition component="DateTime" property="DayOfWeek" condition="lessOrEqual" value=
"5" />
```

```
        </operator>
  </process>
</definition>
```

DateTime component is supported by all operations.

DnsInfo

DnsInfo with the following properties:

Property	Type	Description	Example
DnsQuery	String	The DNS query	www.google.com
DnsQueryLength	Int	The length of the DNS query	100
DnsResponseIpAddressV4	Set of IP v4 addresses	The A records in the DNS response	216.58.201.67
DnsResponseIpAddressV6	Set of IP v6 addresses	The AAAA records in the DNS response	2001:db8:85a3:8131:4321:8a2e:370:7334
DnsResponseString	Set of strings	The NS, CNAME, PTR or TXT records in the DNS reponse	mobile-google.com
DnsResponseLength	Int	The total length of the DNS reponse	100
DnsResponseType	Set of strings	The types of records present in the DNS response	HINFO

For property **DnsResponseType**, there is a pre-defined scope of values that can be used (case insensitive): A, NS, MD, MF, CNAME, SOA, MB, MG, MR, NULL, WKS, PTR, HINFO, MINFO, MX, TXT, RP, AFSDB, X25, ISDN, RT, NSAP, NSAPPTR, SIG, KEY, PX, GPOS, AAAA, LOC, NXT, EID, NIMLOC, SRV, ATMA, NAPTR, KX, CERT, A6, DNAME, SINK, PT, PL, DS, SSHFP, IPSECKEY, RRSIG, NSEC, DNSKEY, DHCID, NSEC3, NSEC3PARAM, TLSA, SMIMEA, HIP, NINFO, RKEY, TALINK, CDS, CDNSKEY, OPENPGPKEY, CSYNC, ZONEMD, SPF, UINFO, UID, GID, UNSPEC, NID, L32, L64, LP, EUI48, EUI64, TKEY, TSIG, IXFR, AXFR, MAILB, MAILA, ANY, URI, CAA, AVC, DOA, AMTRELAY, TA, DLV. For an explanation of these values, follow Resource Record (RR) TYPEs.

Conditions supported for set types (set of IP v4 addresses, set of IP v6 addresses, set of strings) are: contains, notcontains, isempty, isnotempty, isset, isnotset.

Example:

```
<rule>
```

```
    <definition>

        <operations>

            <operation type="DnsRequest">

                <condition component="DnsInfo" condition="contains" property="DnsRes
ponseType" value="CNAME" />

            </operation>

        </operations>

    </definition>

</rule>
```

Supported Operations and their components:

	DnsInfo
DnsRequest	X

Endpoint

Allows you to trigger a rule based on events from client-side antivirus.

Property	Type	Description	Example
DetectionType	String	Detection type	Possible values are: • UnknownAlarm • RuleActivated • MalwareFoundOnDisk • MalwareFoundInMemory • ExploitDetected • FirewallDetection • HipsDetection • BlockedAddress • CryptoBlockerDetection
Scanner	String	Name of the scanner that triggered the event	
Severity	String	Severity of the detection	Possible values are: • Information • Warning • Threat
ThreatHandled	Bool	Information if the threat has been handled	true/false
ThreatName	String	Name of the threat	
ThreatType	String	Type of the threat	Possible values are: • Malware • Nearmiss • PUA • DangerousApp • BlockedFile • UnsafeApp

Supported Operations and their components:

	Endpoint
Detection	X

Enterprise

Provides a statistic about modules in the ESET Inspect monitored network

Property	Type	Description	Example
Safe	Bool	The file is marked as safe	true/false

Supported Operations and their components:

	Module
CreateProcess	X
LoadDLL	X
CodeInjection	X

EnterpriseInspector

Provides information about ESET Inspect Server.

Property	Type	Description	Example
VersionString	String	Version string of ESET Inspect	1.9.2385
VersionMajor	Int	Major version of ESET Inspect	1
VersionMinor	Int	Minor version of ESET Inspect	9
BuildNumber	Int	Build number of ESET Inspect	2385

Applies to:

All operations.

FileAttribute

FileAttribute events are currently reported only on Linux. The rule is triggered when an attribute is set, for example, by the chmod command.

Property	Type	Description	Example
Attribute	String	Attribute of a file	Possible values are: • SUID • SGID • Sticky • Immutable • AppendOnly • Undeletable • Executable

Supported Operations and their components:

	FileAttribute
SetFileAttribute	X

FileItem/DestFileItem

Return the information about the current file

Property	Type	Description	Example
FileNameWithoutExtension	String	Filename without the file extension	*C:\windows\system32\notepad.exe* -> notepad
Extension	String	The file extension	*C:\windows\system32\notepad.exe* -> exe
Path	Path	The file path	*C:\windows\system32\notepad.exe* -> *C:\windows\system32\*
FullPath	Path	The file path including filename	*C:\windows\system32\notepad.exe* -> *C:\windows\system32\notepad.exe*
FileName	String	The filename with the file extension	*C:\windows\system32\notepad.exe* -> notepad.exe
NameLength	Int	The length of the name	*C:\windows\system32\notepad.exe* -> 7
ADS	String	The ADS part of the path	*C:\windows\system32\notepad.exe:example* -> example
isSelf	Bool	Triggers if the operation is done by the file on itself (common for malware to delete itself)	true/false

DestFileItem has the same properties as FileItem, used mostly in combination with FileItem.

Canary File

Path properties have a special variable for Canary files. The value to specify the path to the Canary file is *%CanaryFile%*.

```
<definition>

  <operations>

    <operation type="WriteFile">

      <condition component="FileItem" property="Path" condition="is" value="%CanaryF
ile%" />

    </operation>

    <operation type="RenameFile">

      <condition component="FileItem" property="Path" condition="is" value="%CanaryF
ile%" />

    </operation>

  </operations>

</definition>
```

Supported Operations and their components:

	FileItem	DestFileItem
CodeInjection	X	
CreateNamedPipe	X	
CreateProcess	X	
DeleteFile	X	
LoadDLL	X	
LoadDriver	X	
ModuleDrop	X	
OpenProcess	X	
ReadFile	X	
RenameFile	X	X
SetFileAttribute	X	
TruncateFile	X	
WmiExecution	X	
WriteFile	X	

InspectDetection

InspectDetection is used in Incident Rules to specify a identifier of a certain rule or group of rules that were triggered.

Property	Type	Description	Example
RuleCategory	String	Matches the rule by the category	File system
RuleGuid	String	Matches the rule by the GUID	b7ddfd8b-eb96-4f9e-a3fe-1517aa653b0d
RuleName	String	Matches the rule by the name	F1006
RuleSeverity	Int/Symbols	Matches the rule by the severity	Possible values are: • Information—1 • Warning—2 • Threat—3

Supported Operations and their components:

	InspectDetection
Detection	X

LiveGrid

ESET LiveGrid is a preventative system that gathers information about threats from users worldwide. The LiveGrid database contains reputation information about potential threats. The reputation of executables helps you to filter rule results.

Property	Description	Example
Age	The number of days since the executable was first seen in LiveGrid. The number is rounded to the equivalent of the week, month, half of the year, year, etc.	
Reputation	The number on the reputation scale. The higher, the more trusted	Possible values are: • Trusted – 8, 9 • OK – 6, 7 • Risky – 4, 5 • PUA, Unknown (not seen by LiveGrid) – 3 • Malware – 1, 2
Popularity	The number of computers on which LiveGrid has seen an executable. It is rounded to numbers like 10, 100, 1000, 10000, etc. Malware usually does not exceed the popularity of 1000 until it is detected.	

You cannot rely on LiveGrid values for new executables. Even the most popular and trusted executable (e.g., installer of a new version of Google Chrome) has low popularity and an unknown reputation for some time after its release.

Supported Operations and their components:

	Module
CreateProcess	X
LoadDLL	X
CodeInjection	X

Module

Return the information about the current module

Property	Type	Description	Example
SignerName	String	Name of the signer, if any	"Microsoft Windows"
CompanyName	String	From version info, name of the company that produced the file	"Microsoft Corporation"
FileDescription	String	From version info, file description shown to users	"Microsoft Windows Resource Leak Diagnostic"
FileOrigin	Int/Symbols	File delivered through RDP	Possible values are: • **RDP**—0
ProductName	String	From version info, name of the product with which the file is distributed	"Microsoft Windows Operating System"
FileVersion	String	From version info, the version number of the file	"10.0.14393.0"
ProductVersion	String	From version info, the version number of the product with which the file is distributed	"10.0.14393"
InternalName	String	From version info, internal name of the file	"RdrLeakDiag.exe"
OriginalFileName	String	From version info, original name of the file	"RdrLeakDiag.exe"
PackerName1	String	Name of the packer	"UPX"
SFXName	String	Name of the sfx packer	"Zip"
Sha1	Hash	sha1 hash of the executable	fa7ebffd41bc44c47ea1b11928ee368c19f6d6a2

Property	Type	Description	Example
MD5	Hash	md5 hash of the executable	
Sha256	Hash	sha256 hash of the executable	
SignatureType	Int/Symbols	Signature type of the executable	Possible values are: • **Trusted**—90—the signature is trusted by Endpoint • **Valid**—80—the signature is trusted by the OS • **Adhoc**—75—the certificate is self signed • **None**—70—there is no signature in the file • **Invalid**—60—the signature is not valid/corrupted/revoked • **Unknown**—50—failed to verify certificate • **Present**—50—the signature is present, but the certificate status is unknown
Whitelist	Int/Symbols	Whitelist type of the executable	Possible values are: • **None**—no whitelisting for this file • **Authoritative**—the file is whitelisted by EndPoint • **LiveGrid**—the file is whitelisted from LiveGrid • **Certificate**—the file certificate is whitelisted
EmulationStatus	Int	The status of the file emulation (if the file was emulated by advanced heuristics)	0—Was not emulated 1—Was emulated
FileSize	Long	Filesize in bytes	41984
IsElf	Bool	The file is an ELF file	true/false
IsExe	Bool	The file is a Windows executable	true/false
IsDLL	Bool	The file is a PE DLLs	true/false
IsNative	Bool	The file is a native PE executable	true/false
DaysSinceLastNearMiss	Int	Number of days since the file was recognized as nearmiss. Nearmiss—the detection is triggered due to malware, but it may be a false positive (we cannot guarantee it is malware)	
MachoSignatureId	String	Identifier of a Mach-O file present in the signature	"com.apple.ls"
IsMacho	Bool	Defines whether a file is a Mach-O (macOS) file or not	

Property	Type	Description	Example
MachoUserId	String	Unique developer ID assigned by Apple	
MachoSignerCns	String	Set of common names from certificates in Mach-O file	
MachoIsProtected	Bool	Module is a protected Mach-O executable	
Tags	String	Allows a user to filter by a module that has a specified tag attached	

1Names of packers may change in the future. Therefore we recommend using isnotempty or isempty value for the condition.

Supported Operations and their components:

	Module
CreateProcess	X
LoadDLL	X
CodeInjection	X

Network

Return information about network events

Property	Type	Description	Example
DestinationIpAddressV4	ipv4 address	The ipv4 destination address of Firewall detection. Supports masks.	192.168.0.1, supports masks - 192.168.0.0/16
DestinationIpAddressV6	ipv6 address	The ipv6 destination address of Firewall detection. Supports masks.	2001:db8:85a3:8d3:1319:8a2e:370:0, supports masks - 2001:db8:85a3:8d3:1319:8a2e:370:0/112
Hostname	String	The target hostname	www.google.com
Inbound	Bool	The connection is inbound	true/false
IpAddressV4	ipv4 address	The ipv4 address target of the event. Supports masks.	192.168.0.1, supports masks - 192.168.0.0/16
IpAddressV6	ipv6 address	The ipv6 address target of the event. Supports masks.	2001:db8:85a3:8d3:1319:8a2e:370:0, supports masks - 2001:db8:85a3:8d3:1319:8a2e:370:0/112
Port	Int	The TCP/UDP target port	8080

Property	Type	Description	Example
Protocol	String	The protocol used by the connection	HTTP, HTTPS, etc.
SourceIpAddressV4	ipv4 address	The ipv4 source address of Firewall detection. Supports masks.	192.168.0.1, supports masks - 192.168.0.0/16
SourceIpAddressV6	ipv6 address	The ipv6 source address of Firewall detection. Supports masks.	2001:db8:85a3:8d3:1319:8a2e:370:0, supports masks - 2001:db8:85a3:8d3:1319:8a2e:370:0/112
Url	String	If the request involved a URL (i.e., HTTP request)	The target URL

Example:

```
<definition>

 <operations>

         <operation type="TcpIpConnect">

                 <operator type="or">


 <condition component="Network" property="IpAddressV4" condition="is" value="10.0.0.
0/8" />


 <condition component="Network" property="IpAddressV4" condition="is" value="172.16.
0.0/12" />


 <condition component="Network" property="IpAddressV4" condition="is" value="192.168
.0.0/16" />


 <condition component="Network" property="IpAddressV4" condition="is" value="127.0.0
.0/8" />


 <condition component="Network" property="IpAddressV6" condition="is" value="::1/128
" />


 <condition component="Network" property="IpAddressV6" condition="is" value="fc00::/
7" />

                 </operator>

         </operation>
 </operations>

</definition>
```

Supported Operations and their components:

	Network
Detection	X
HttpRequest	X
TcpIpAccept	X
TcpIpConnect	X
TcpIpProtocolIdentified	X

OpenProcess

Added a new rule attribute, which triggers when a process is opened.

HIPS sends OpenProcess events only for lsass.exe and only with PROCESS_VM_WRITE and/or PROCESS_VM_READ process access only when calling OpenProcess or DuplicateHandle (when the already opened process with mentioned accesses)

Properties are:

AccessRight—it can have these values PROCESS_VM_WRITE, PROCESS_VM_READ

Example:

```
<operations>

    <operation type="OpenProcess">

        <condition component="OpenProcess" property="AccessRight" condition="contain
s" value="PROCESS_VM_READ" />

    </operation>

</operations>
```

Supported Operations and their components:

	OpenProcess
OpenProcess	X

ProcessBehavior

ProcessBehavior events are reported on Windows. Antivirus tracks which files were accessed and how frequently. If the process accessed more files than are set in the barrier in a certain time period, this event is reported.

Property	Type	Description	Example
FileOperations	String	Operations and their barriers	Possbile values are: • WrBarrier1 • WrBarrier2 • RenameBarrier1 • RenameBarrier2 • MultiextWrBarrier1 • MultiextWrBarrier2 • MultiextRenameBarrier1 • MultiextRenameBarrier2 • BlobDetection

The **FileOperations** values are internal and are intended to be used only with default rules.

Supported Operations and their components:

	ProcessBehavior
MultipleFilesChanged	X

ProcessInfo

Return information about the current process

Property	Type	Description	Example
CommandLine	String	Process command line	file.txt
CommandLineLength	Int	Length of the command line	123
Compromised	Bool	The process was marked as compromised by a rule with MarkAsCompromised action	true/false or 1/0
IntegrityLevel	Int/Symbols	Integrity level of the process	Possible values are: • Untrusted—0 • Low—4096 • Medium—8192 • High—12288 • System—16384 • Protected process—20480
LnkPath	String	Contains a path to a shortcut execution	
ProcessLevel	Int	Depth of the process in process hierarchy	123
ProcessDistance	Int	The distance of the process from the current process	123
ProcessOwner	String	The user that created the process	

Property	Type	Description	Example
CaseSensitiveCommandLine	String	Allows creating rules for command line that is case sensitive	

Supported Operations and their components:

	Module
CreateProcess	X
LoadDLL	X
CodeInjection	X

RegistryItem

Return the information about registry events

Property	Type	Description	Example
Key	Path	The path of the key, contains the Value Name of the key	HKLM\SOFTWARE\ESET\EnterpriseInspector\EIAgent\CurrentVersion\Info
ValueLength	Int	The length of the key value	16
StringValue	String	The value of a string key	"ExampleValue"
IntValue	Int	The value of int key	32

Example of use

For registry key: "*HKLM\SOFTWARE\Classes\CLSID\{12788EFC-0553-4126-A4E1-8AA0F5270615}\InprocServer32\CodeBase*"

```
<operations>
  <operation type="RegSetValue">
     <operator type="AND">
         <condition component="RegistryItem" property="Key" condition="starts" value
="HKLM\software\classes\clsid\" />
         <condition component="RegistryItem" property="Key" condition="contains" val
ue="\InprocServer32\" />
         <condition component="RegistryItem" property="Key" condition="ends" value="
\CodeBase" />

         <condition component="RegistryItem" property="StringValue" condition="conta
ins" value="http" />
      </operator>
```

```
   </operation>
</operations>
```

Supported Operations and their components:

	RegistryItem
RegSetValue	X
RegDeleteKey	X
RegDeleteValue	X
RegRenameKey	X

Scripts

This component has two properties:

Property	Type	Description
Script	String	Script received by ESET PROTECT via AMSI intergration
ScriptLength	Int	Length of the script (count of characters)
ScriptSha1	Hash	SHA1 hash of the script fragment
ScriptSha256	Hash	SHA256 hash of the script fragment

Example:

```
<rule>

    <definition>

        <operations>

            <operation type="Scripts">

                <condition component="Scripts" property="ScriptLength" condition="gr
eaterOrEqual" value="15" />

            </operation>

        </operations>

    </definition>

    <description>

        <name>amsiTriggerScriptLength</name>

        <category>TEST</category>

    </description>

</rule>
```

Supported Operations and their components:

	Scripts
Scripts	X

SystemInfo

Info about the system the events are coming from.

Property	Type	Description	Example
SystemType	Int	What type of operating system triggered the event	
SystemVersion	String	The whole system version with major and minor numbers	
SystemArchitecture	Int	The type of system architecture	For example, 64-bit
Tags	Set of strings	Tags assigned to that specific system	

Applies to:

All operations.

TargetUser/DoneByUser

The user that was targeted by the DoneByUser.

Property	Type	Description	Example
UserName	String	Which user (account name) is behind the given event	
Sid	String	Which user (account SID) is behind the given event	
SidNameUse	Int	Which group (its SID type) is behind the given event	Groups are: • "User" • "Group" • "Domain" • "Alias" • "WellKnownGroup" • "DeletedAccount" • "Invalid" • "Unknown" • "Computer" • "Label" • "LogonSession"

DoneByUser has the same properties as TargetUser.

Supported Operations and their components:

	TargetUser/DoneByUser
UserActivate	X
UserAddToGroup	X
UserCreate	X
UserDisable	X
UserLogin	X
UserLogout	X
UserRemove	X
UserRemoveFromGroup	X

UserGroupData

Same as UserLogonData but with properties to data about groups.

Properties are:

Property	Type	Description	Example
Sid	String	Which group (group SID) is behind the given event	
GroupName	String	Which group (group name) is behind the given event	
SidNameUse	Int	Which group (its SID type) is behind the given event	Groups are: • "User" • "Group" • "Domain" • "Alias" • "WellKnownGroup" • "DeletedAccount" • "Invalid" • "Unknown" • "Computer" • "Label" • "LogonSession"

Supported Operations and their components:

	UserGroupData
UserActivate	X
UserAddToGroup	X
UserCreate	X
UserDisable	X
UserLogin	X
UserLogout	X
UserRemove	X
UserRemoveFromGroup	X

UserLogonData

Events related to the User activities (created user, logged in, ...), information about those events.

Property	Type	Description	Example
LogonType	Int	Type of login	Symbols and values are: • "Unknown" • "Interactive" • "Network" • "Batch" • "Service" • "Unlock" • "NetworkCleartext" • "NewCredentials" • "RemoteInteractive" • "CachedInteractive"

Supported Operations and their components:

	UserLogonData
UserLogin	X
UserLogout	X

WmiExecutionInfo

WMI execution event occurs only when the WMI method, Win32_process.create() is called.

Property	Type	Description
MethodName	String	A method that was triggered
ClassName	String	A class containing a triggered method
CommandLine	String	A command line sent to a method as a list of arguments
IsLocal	Bool	Determines if a method was called locally or remotely

Example:

```
<rule>

    <definition>

        <operations>

            <operation type="WmiExecution" >

                <condition component="WmiExecutionInfo" property="CommandLine" condi
tion="is" value="notepad.exe"/>
```

```
            </operation>

        </operations>

    </definition>

    <description>

        <name>WMI Execution event where argument is notepad.exe</name>

        <category>Default</category>

    </description>

</rule>
```

Supported Operations and their components:

	WmiExecutionInfo
WmiExecution	X

WmiPersistenceInfo

The event is generated when consumer binds to a filter.

Property	Type	Description
EventFilterName	String	Name of the used EventFilter
EventConsumerName	String	A name of a consumer which triggers an action when a specific event arrives
Handler	String	Command line executed by an event consumer
Query	String	A query in an event filter that captures events that should execute an action
TriggeringUserName	String	A name of a user who triggered an event matched by a filter
TriggeringUserSid	String	Triggering the user's security ID
TriggeringUserSidNameUse	Int	Triggering the user's SID type

Example:

```
<rule>

    <definition>

        <operations>

            <operation type="WmiPersistence" >

                <condition component="WmiPersistenceInfo" property="TriggeringUserNa
me" condition="is" value="domain\user.name"/>

            </operation>
        </operations>

    </definition>

    <description>

        <name>WMI Persistence event triggered by user.name</name>

        <category>Default</category>

    </description>

</rule>
```

Supported Operations and their components:

	WmiPersistenceInfo
WmiPersistence	X

WmiQueryInfo

WMI query events occur when a user or a service trigger a query on a system.

Property	Type	Description
Query	String	A query was triggered in a system
IsLocal	Bool	If false, a query was called from a remote machine (for example, using WbemTest)

	WmiQueryInfo
WmiQuery	X

Example event:

```
<?xml version="1.0" encoding="utf-8"?>

<rule>

    <definition>

        <operations>

            <operation type="WmiQuery">

                <condition component="WmiQueryInfo" property="Query" condition="cont
ains" value="win32_service" />

            </operation>

        </operations>
    </definition>

    <description>

        <name>Example WMI query event</name>

        <explanation>

            This tag supports markdown and html syntax.

            It is also true for maliciousCauses, benignCauses and recommendedActions
 tags.

        </explanation>

        <maliciousCauses>

            Content of tags with HTML text must be surrounded with CDATA xml tag.

        </maliciousCauses>

        <category>

            Default

        </category>

    </description>

</rule>
```

Property Types & Relations, Symbols

Property types & Relations (condition attribute).

	is(not)set	is(not)	is(not)empty	(not)starts	(not)contains	(not)ends	less, lessOrEqual, greater, greaterOrEqual
string	✔	✔	✔	✔	✔	✔	
int	✔	✔			✔		✔
value	✔	✔	✔				
bool	✔	✔					
date	✔	✔					✔
set of strings	✔		✔		✔		
IPv4 Address	✔	✔	✔				
IPv6 Address	✔	✔	✔				
set of IPV4 addresses	✔		✔		✔		
set of IPV6 addresses	✔		✔		✔		

Symbols

When specifying a value for a property to be matched against:

```
<condition component="FileItem" property="FileNameWithoutExtension" condition="is"
value="svchost">
```

(the "svchost" string), for certain properties, you can use a `value` from pre-defined symbols (to avoid having to specify integer constants that can/will be modified), currently the following are implemented:

- Module::WhiteList
  - None—no whitelisting for this file
  - Authoritative—the file is whitelisted by EndPoint
  - LiveGrid—the file is whitelisted from LiveGrid
  - Certificate—the file certificate is whitelisted
- Module::SignatureType
  - Trusted—90—the signature is trusted by Endpoint
  - Valid—80—the signature is trusted by the OS
  - Adhoc—75—the certificate is self signed
  - None—70—there is no signature in the file
  - Invalid—60—the signature is not valid/corrupted/revoked
  - Unknown—50—failed to verify certificate
  - Present—50—the signature is present, but the certificate status is unknown
- ProcessInfo::IntegrityLevel
  - Untrusted—0
  - Low—4096
  - Medium—8192
  - High—12288
  - System—16384
  - Protected process—20480
- SystemInfo::SystemType
  - Windows

    - Win
    - Apple
    - macos
    - macosx
    - osx
- SystemInfo::SystemArchitecture
    - 32
    - 32bit
    - x86
    - 64
    - 64bit
    - x64
    - amd64
- <Whatever>::SidNameUse
    - "User"
    - "Group"
    - "Domain"
    - "Alias"
    - "WellKnownGroup"
    - "DeletedAccount"
    - "Invalid"
    - "Unknown"
    - "Computer"
    - "Label"
    - "LogonSession"
- UserLogonData::LogonType
    - "Unknown"
    - "Interactive"

o"Network"

o"Batch"

o"Service"

o"Unlock"

o"NetworkCleartext"

o"NewCredentials"

o"RemoteInteractive"

o"CachedInteractive"

For LogonType definition, see.

- CodeInjection::CodeInjectionType

  oCreateRemoteThread

  oSetThreadContext

  oApcQueue

For example, for ProcessInfo component and IntegrityLevel property:

```
<condition component="ProcessInfo" property="IntegrityLevel" condition="is" value="L
ow">
```

Supported environment variables

Use the following variables in the rules if you want to match a specific system path. These variables substitute the system path of an event being executed on a client machine. Only events using such variables will be processed by a rule. Therefore, *c:\windows\system32* and also *%WINDIR%\system32* will not be matched, but %SYSTEM% will.

Windows

%SYSTEM%	`%SYSTEMDRIVE%\windows\system32\`
%WINDIR%	`%SYSTEMDRIVE%\windows\`
%PROGRAMDATA%	`%SYSTEMDRIVE%\programdata\`
%PROGRAMFILES%	`%SYSTEMDRIVE%\program files\`
%PROGRAMFILES(X86)%	`%SYSTEMDRIVE%\program files (x86)\`
%APPDATA%	`%SYSTEMDRIVE%\users\*\appdata\roaming\`
%LOCALAPPDATA%	`%SYSTEMDRIVE%\users\*\appdata\local\`
%HOME%	`%SYSTEMDRIVE%\users\*\`
%TMP%	`%SYSTEMDRIVE%\users\*\appdata\local\temp\`

%SYSTEM%	%SYSTEMDRIVE%\windows\system32\
HKCU	REGISTRY ONLY! Computer\HKEY_CURRENT_USER\
HKLM	REGISTRY ONLY! Computer\HKEY_LOCAL_MACHINE\
%RemovableDrive%	Points to place on any removable drive
%RemoteDrive%	Points to place on any remote drive
%CDROM%	Points to place on any CD-ROM drive
%COMMONAPPDATA%	%ALLUSERSPROFILE%
%COMMONDESKTOP%	%PUBLIC%\desktop\
%COMMONDOCUMENTS%	%PUBLIC%\documents\
%COMMONPROGRAMS%	%ALLUSERSPROFILE%\microsoft\windows\start menu\programs\
%COMMONSTARTMENU%	%ALLUSERSPROFILE%\microsoft\windows\start menu\
%COMMONSTARTUP%	%ALLUSERSPROFILE%\microsoft\windows\start menu\programs\startup\
%COMMONTEMPLATES%	%ALLUSERSPROFILE%\microsoft\windows\templates\
%COMMONMUSIC%	%PUBLIC%\music\
%COMMONPICTURES%	%PUBLIC%\pictures\
%COMMONVIDEO%	%PUBLIC%\video\
%STARTMENU%	%SYSTEMDRIVE%\users\*\appdata\roaming\microsoft\windows\start menu\
%STARTUP%	%SYSTEMDRIVE%\users\*\appdata\roaming\microsoft\windows\start menu\programs\startup\
%DESKTOP%	%SYSTEMDRIVE%\users\*\desktop\
%LOCALAPPDATALOW%	%SYSTEMDRIVE%\users\*\appdata\locallow\
%TEMP%	%SYSTEMDRIVE%\users\*\appdata\local\temp\
%SYSTEMDRIVE%	usually "C:"
%ALLUSERSPROFILE%	= %PROGRAMDATA% = c:\programdata
%PUBLIC%	c:\users\public

Apple

%APPLICATIONS%	**/applications/**
%COMMONSTARTUPADMIN%	/library/startupitems/
%COMMONSTARTUPOS%	/system/library/startupitems/
%DESKTOPMAC%	~/desktop/
%DOCUMENTSMAC%	~/documents/
%DOWNLOADSMAC%	~/downloads/
%HOME%	~/
%LIBRARY%	/library/
%LIBRARYAPPSUPPORT%	/library/application support/
%LIBRARYEXTENSIONS%	/library/extensions/
%LIBRARYKEYCHAINS%	/library/keychains/
%LIBRARYPREFERENCES%	/library/preferences/

%APPLICATIONS%	/applications/
%VOLUMES%	/volumes/
%MOVIES%	~/movies/
%MUSICMAC%	~/music/
%NET%	/net/
%PICTURESMAC%	~/pictures/
%PROCSTARTBOOTBYADMIN%	/library/launchdaemons/
%PROCSTARTBOOTBYOS%	/system/library/launchdaemons/
%PROCSTARTUSERBYADMIN%	/library/launchagents/
%PROCSTARTUSERBYOS%	/system/library/launchagents/
%PROCSTARTUSERBYUSER%	~/library/launchagents/
%PUBLIC%	~/public/
%SYSTEMLIBRARY%	/system/library/
%SYSTEMLIBRARYEXTENSIONS%	/system/library/extensions/
%SYSTEMLIBRARYPREFERENCES%	/system/library/preferences/
%TMPMAC%	/tmp/
%TMPDIRVAR%	/var/folders and /private/var/folders
%TMPLIBRARY%	/library/caches/
%TMPLOCALLIBRARY%	~/library/caches/
%TMPPRIVATE%	/private/tmp/
%USERLIBRARY%	~/library/
%USERLIBRARYAPPSUPPORT%	~/library/application support/
%USERLIBRARYKEYCHAINS%	~/library/keychains/
%USERLIBRARYPREFERENCES%	~/library/preferences/
%USERSMAC%	/users/

Example of use

```
<process>
    <operator type="AND">
        <condition component="FileItem" property="Path" condition="is" value="%AppDa
ta%\Roaming\" />
        <condition component="FileItem" property="Extension" condition="is" value="e
xe" />
    </operator >
</process>
```

Best Practices

- Test new rules in a test environment first or on a smaller set of computers.
- Avoid designing rules that produce many alerts, such as "any process was started".

- When creating a new rule, document what and why is monitored.

- Define rule severity when creating a new rule. If the `<severity>` tag is not present, the rule has the Warning severity automatically assigned, which may not fit the specific rule.

- ESET Inspect does not use CurrentControlSet registry key, as this key is an alternating symbolic link that is dynamically evaluated by the operating system and is pointing to ControlSet%number%. See how to match registry keys/values inside CurrentControlSet in Rules Examples.

- Because of how the x86 emulation works on x64 Windows OS, many registry keys/values also have their counterpart under Wow6432Node with similar functionality present, so you need to monitor this location. The same concept applies to %windir%\SysWOW64 and %PROGRAMFILES(X86)% folders.

- The best way to match a specific registry value is to use the condition value `ends` because of how the registry paths are implemented in Windows.

- The best way to match alternate data streams (ADS) on Windows NTFS is to use the following condition:
```
<condition component="FileItem" property="Extension" condition="contains" value=":"
/>
```

The "`contains`" comparator is quite heavy on performance. If possible, use "`starts`" or "`ends`" instead.

- You can use a special value condition "isnotempty", which indicates that detection should be triggered on any value—useful, for example, to match any network connection from a specific process. Example use case:
```
<condition component="FileItem" property="FileNameWithoutExtension"
condition="isnotempty" />
```

- Registry hives are matched via their shortened names, specifically HKCU for HKEY_CURRENT_USER and HKLM for HKEY_LOCAL_MACHINE.

- Condition paths are matched case-insensitive.

- When creating a rule, plan how to filter false positives (too many irrelevant alerts). Generally speaking, in the first iteration, you should be as general as possible (for example, any change of this registry value) and only add filters subsequently (for example, if the general rule produces many irrelevant detections). Afterward, you can add filters (for example, LiveGrid Popularity/Reputation, Process Name) to reduce the number of irrelevant detections. Filters should be as specific as possible to not lose relevant or noteworthy detections.

- Short-circuit evaluation of logical operators (tag <operator>) is implemented, so when creating the rule, you can consider optimization of rule logical expression to improve rule matching performance. Practical example—logical expression (A | B) & C can be rewritten as C & (A | B), and if C is not true, the rest of the expression will not be evaluated.

Rules Examples

- Working with registry
- Monitoring network connections
- Working with URLs
- Working with command line

- Working with a parent-child relationship
- Working with LiveGrid and Safe property
- Working with compromised flag

Working with registry

We want to monitor changes made to registry value AppInit_DLLs that allows automatic loading of dynamic-link library (DLL) to certain processes on the system. A related registry value with similar functionality is AppCertDlls. Whole registry value paths are:
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs
HKLM\SYSTEM\CurrentControlSet\Control\SESSION MANAGER\ AppCertDlls

Rule

```
<?xml version='1.0' encoding='UTF-8'?>

<rule>

  <description>

    <name>AppInit DLL Registry Creation [A0101]</name>

    <category>Persistence</category>

    <os>Windows</os>

    <severity>80</severity>

    <mitreattackid>T1218.011</mitreattackid>

    <explanation>AppInit DLL is a mechanism that allows an arbitrary list of DLLs to
 be loaded into each user-
mode process on the system. DLLs that are specified in the `AppInit_DLLs` value in t
he Registry key `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Win
dows` are loaded by user32.dll into (almost) every process that loads user32.dll. Th
e AppInit DLL functionality is disabled in Windows 8 and later versions when secure
boot is enabled.</explanation>

    <benignCauses>AppInit_DLLs are rarely used by specific software, such as graphic
 card support dlls or virtual machine software.</benignCauses>

    <maliciousCauses>AppInit_DLLs are sometimes used by malware to achieve persisten
ce on the target machine.</maliciousCauses>

    <recommendedActions>1. Evaluate if the change to the Applnit_DLLs correlates wit
h known software, a software update, patch cycles, etc.

2. Evaluate the process/module that made the change.

3. Check for presence of new/non-standard DLLs on the computer.

4. If a suspicious process/module or DLL is detected, start the incident response pr
ocess (for example, disconnect the computer from the internet, update your antivirus
 product and scan the computer for malware, send samples for analysis, block modules
, etc.).</recommendedActions>

  </description>

  <definition>

    <operations>

      <operation type="RegSetValue">

        <operator type="OR">

          <operator type="AND">

            <operator type="OR">

              <condition component="RegistryItem" property="Key" condition="starts"
value="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows"/>

              <condition component="RegistryItem" property="Key" condition="starts"
value="HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows"/>

            </operator>

            <condition component="RegistryItem" property="Key" condition="ends" valu
e="Appinit_Dlls"/>

          </operator>

          <operator type="AND">

            <condition component="RegistryItem" property="Key" condition="starts" va
lue="HKLM\SYSTEM\ControlSet"/>

            <condition component="RegistryItem" property="Key" condition="ends" valu
e="Control\SESSION MANAGER\AppCertDlls"/>

          </operator>

        </operator>

      </operation>

    </operations>

  </definition>

  <maliciousTarget name="current"/>

  <actions>

    <action name="TriggerDetection"/>

    <action name="StoreEvent"/>

  </actions>

</rule>
```

Things to notice in the rule example above:

1.Use of shortened HKEY values instead of full ones because full HKEY values are not matched.

2.Inclusion of Wow6432Node for AppInit_DLLs. This and many other values are duplicated in this registry key for x86 support on x64 systems. We also need to monitor this value.

3.Matching registry value via `ends` condition. Firstly, we decided to match the registry value name "AppInit_DLLs" and check if the path to the registry value is the wanted one. This approach should theoretically lower the server's workload because of the short-circuit evaluation of conditions. Using the whole registry value path for matching ("*HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs*") is also acceptable.

4.CurrentControlSet registry value is an alternating symbolic link that is dynamically evaluated by the operating system and is pointing to ControlSet%number%. As Enterprise Inspector receives a registry path with ControlSet%number% value, we split the registry value path into two components.

Monitoring network connections

Rundll32 is a Microsoft Windows system utility that provides an entry point and minimal framework for executing dynamic-link libraries (DLL). We want to monitor any network connections made by this utility.

Rule

```
<?xml version='1.0' encoding='UTF-8'?>

<rule>

  <description>

    <name>External Network Connection from rundll32.exe with Unpopular Parent [A0504
b]</name>

    <category>Communication</category>

    <os>Windows</os>

    <severity>48</severity>

    <mitreattackid>T1218.011</mitreattackid>

    <explanation>Rundll32 is a Microsoft Windows system utility that provides an ent
ry point and minimal framework for executing dynamic load libraries. The rule monito
rs network connections to public range IP addresses from Rundll32 that was started f
rom an unpopular process.</explanation>

    <benignCauses>This usually happens when benign less popular software prints on t
he network printer.</benignCauses>

    <maliciousCauses>Often used by script malware for downloading or reporting</mali
ciousCauses>
```

```
    <recommendedActions>1. Evaluate the parent process, its command line and executi
on chain.

2. Evaluate the target IP, check events for creating, modifying and executing files
by the script interpreter.

3. If a suspicious activity is detected, start the incident response process (for ex
ample, disconnect the computer from the internet, update your antivirus product and
scan the computer for malware, send samples for analysis, block module, etc.).</reco
mmendedActions>

  </description>

  <definition>

    <parentprocess>

      <operator type="AND">

        <condition component="LiveGrid" property="Popularity" condition="less" value
="1000"/>

        <operator type="NOT">

          <operator type="OR">

            <condition component="Module" property="SignatureType" condition="is" va
lue="Trusted"/>

            <condition component="Enterprise" property="Safe" condition="is" value="
1"/>

          </operator>

        </operator>

      </operator>

    </parentprocess>

    <process>

      <operator type="OR">

        <condition component="FileItem" property="FileName" condition="is" value="ru
ndll32.exe"/>

        <condition component="Module" property="OriginalFileName" condition="is" val
ue="RUNDLL32.exe"/>

      </operator>

    </process>

    <operations>

      <operation type="TcpIpConnect">

        <operator type="OR">
```

```
          <operator type="NOT">
            <operator type="OR">
              <condition component="Network" property="IpAddressV4" condition="is" v
alue="10.0.0.0/8"/>
              <condition component="Network" property="IpAddressV4" condition="is" v
alue="172.16.0.0/12"/>
              <condition component="Network" property="IpAddressV4" condition="is" v
alue="192.168.0.0/16"/>
              <condition component="Network" property="IpAddressV4" condition="is" v
alue="127.0.0.0/8"/>
            </operator>
          </operator>
          <operator type="NOT">
            <operator type="OR">
              <condition component="Network" property="IpAddressV6" condition="is" v
alue="fc00::/7"/>
              <condition component="Network" property="IpAddressV6" condition="is" v
alue="::1/128"/>
            </operator>
          </operator>
        </operator>
      </operation>
    </operations>
  </definition>
  <maliciousTarget name="current"/>
  <actions>
    <action name="TriggerDetection"/>
    <action name="StoreEvent"/>
  </actions>
</rule>
```

Things to notice in the rule example above:

1.While testing the rule, we noticed that rule was triggered by printing on a network printer, which is internally handled by Rundll32. As this case is false positive, we decided to filter out Rundll32 utilities started from popular processes – usage of `<parentprocess>`. We could also use other filters, such as

Trusted or Marked as Safe.

2.Rundll32 is matched by its common name because the process executable can be renamed. We can also match Rundll32 using the Executable OriginalFileName property.

3.As we want to monitor network connection, we use **TcpIpConnect** operation.

Working with URLs

Common behavior among malware is downloading additional parts of malware or malware configuration data from publicly available data sharing services such as *pastebin.com*. We want to monitor each access to *pastebin.com*. We need to filter out valid cases, such as a user browsing the internet on purpose, and we can choose to use the `popularity` property.

Rule

```
<?xml version='1.0' encoding='UTF-8'?>

<rule>

  <description>

    <name>Unpopular Process Makes HTTP Request to a PasteBin-
like Site [E0505]</name>

    <category>Communication</category>

    <os>Windows</os>

    <severity>80</severity>

    <mitreattackid>T1102.001</mitreattackid>

    <explanation>Public Web services, including ones like pastebin.com(and similar),
 are typically accessed via web browser applications. The aim of this rule is to try
 catch instances where pastebin-
like sites are accessed by unpopular processes that would likely be considered suspi
cious in the hopes it may highlight instances worth investigating amongst other past
ebin-
like activity. This rule may generate a number of initial false positives before bei
ng tuned</explanation>

    <benignCauses>Legit, unpopular executable is used to contact one of these domain
s; this still warrants investigation to verify.</benignCauses>

    <maliciousCauses>Process making a HTTP request to a PasteBin-
like URL that contains:

            -
 C&amp;C infrastructure information (domains, IPs, commands/instructions, etc.),

            - further malicious payload stages.

            The process could also be exfiltrating data to this site.</maliciousCaus
es>
```

```
    <recommendedActions>1. Evaluate the process tree lineage, its command line and s
urounding events.

2. Evaluate the local host, check events for the creation, modification, and executi
on of suspicious files.

3. Evaluate the other detections from this host to identify related activity.

4. If malicious activity is detected, start your incident response procedures (for e
xample, isolate the computer from the internet, update your antivirus signatures and
 scan the computer for malware, send samples for analysis, block module, etc.).</rec
ommendedActions>

  </description>

  <definition>

    <process>

      <operator type="AND">

        <condition component="LiveGrid" property="Popularity" condition="less" value
="1000"/>

        <operator type="NOT">

          <operator type="OR">

            <condition component="Module" property="SignatureType" condition="is" va
lue="Trusted"/>

            <condition component="Enterprise" property="Safe" condition="is" value="
1"/>

          </operator>

        </operator>

      </operator>

    </process>

    <operations>

      <operation type="HttpRequest">

        <operator type="OR">

          <condition component="Network" property="Url" condition="contains" value="
pastebin.com"/>

          <condition component="Network" property="Url" condition="contains" value="
0bin.net"/>

          <condition component="Network" property="Url" condition="contains" value="
pastie.org"/>

          <condition component="Network" property="Url" condition="contains" value="
pastebin.pl"/>
```

```
          <condition component="Network" property="Url" condition="contains" value="
hastebin.com"/>

        </operator>

      </operation>

    </operations>

  </definition>

  <maliciousTarget name="current"/>

  <actions>

    <action name="TriggerDetection"/>

    <action name="StoreEvent"/>

    <action name="SubmitParentToLiveGuard"/>

  </actions>

</rule>
```

Things to notice in the rule example above:

1. As *pastebin.com* may have different IPs associated, we are matching URL *pastebin.com* directly.

Working with command line

Sometimes, filecoder malware uses a legitimate program to encrypt and delete user files. RAR archiver can be such a program, so we want to create a rule to monitor the execution of the RAR archiver with specific parameters used to encrypt the archive file and delete source files.

Rules

```
<?xml version='1.0' encoding='UTF-8'?>

<rule>

  <description>

    <name>RAR Encrypts and Deletes Files [B0601]</name>

    <category>Ransomware / Filecoders</category>

    <os>Windows</os>

    <severity>84</severity>

    <mitreattackid>T1560.001,T1486</mitreattackid>

    <explanation>A RAR archiver was executed, instructed to password-
protect an archive, and delete the source files. If the user isn't aware of the acti
```

```
vity, it may indicate ransomware activity.</explanation>

    <benignCauses>It can be a legitimate action of the user to protect data by encry
pting it.</benignCauses>

    <maliciousCauses>Used by some filecoders to encrypt and delete a user's data.</m
aliciousCauses>

    <recommendedActions>1. Investigate the activity:

   * Is it a single activity or a sequence?

   * What is the count and type of files involved?

   * Was it initiated by the user or not?

2. What process initiated the activity?

   * Is it a well known and trusted program?

   * Is it signed by a trustworthy vendor?

   * Scan it with your Antivirus product or check its reputation and popularity dire
ctly in the Inspect console.

   * If suspicious, submit the program for further analysis.

3. Use the password from command line logged by Inspect to recover the encrypted fil
es.</recommendedActions>

  </description>

  <definition>

    <process>

      <operator type="AND">

        <condition component="FileItem" property="FileNameWithoutExtension" conditio
n="is" value="rar"/>

        <operator type="AND">

          <operator type="OR">

            <condition component="ProcessInfo" property="CommandLine" condition="con
tains" value="-p"/>

            <condition component="ProcessInfo" property="CommandLine" condition="con
tains" value="-hp"/>

          </operator>

          <operator type="OR">

            <condition component="ProcessInfo" property="CommandLine" condition="con
tains" value="-df"/>

            <condition component="ProcessInfo" property="CommandLine" condition="con
tains" value="-dw"/>
```

```
        </operator>
      </operator>
    </operator>
  </process>
</definition>
<maliciousTarget name="parent"/>
<actions>
  <action name="TriggerDetection"/>
  <action name="StoreEvent"/>
  <action name="SubmitParentToLiveGuard"/>
</actions>
</rule>
```

Things to notice in the rule example above:

1. CommandLine property is used with condition `contains` to select only specific parameters and leave the rest of the command line arbitrary.
2. Conditions are combined with logical operators OR and AND to achieve the desired outcome.

Working with a parent-child relationship

This topic addresses whether malware is delivered as a script in an email attachment or in a document.

We want to create a rule monitoring execution of some sort of script interpreter (executing scripts) originating from Microsoft Office application, that is, some document or email.

Rule

```
<?xml version='1.0' encoding='UTF-8'?>
<rule>
  <description>
    <name>Microsoft Office Application Invoked Script Interpreter [D0807]</name>
    <guid>4e9047f1-c506-4461-a2f3-a4e1db82ce48</guid>
    <category>Office</category>
    <os>Windows</os>
    <severity>77</severity>
    <mitreattackid>T1059.005,T1203</mitreattackid>

    <explanation>Malicious documents are one of the common techniques used for initi
al access. Adversaries commonly abuse features such as Macros or Add-
Ins. For legacy versions of Microsoft Office public exploits are also available. Rul
e monitors following applications: Excel, Access, Outlook, PowerPoint and Word that
are executing Windows Command Prompt, PowerShell, Windows Script Host or Microsoft H
TML Application.</explanation>

    <benignCauses>Custom Office documents usually used for automation of tasks such
as internal asset management. Various Legal Management Software suites that integrat
e with Microsoft Office can trigger this behavior.</benignCauses>

    <maliciousCauses>Malicious Microsoft Office document.</maliciousCauses>

    <recommendedActions>1. Investigate the process tree for any additional detection
s indicating suspicious activity.

2. Check the results of document analysis in ESET LiveGuard if available.

3. Investigate other detections on the same host.

4. Initiate the incident response process based on investigation outcome.</recommend
edActions>

  </description>

  <definition>

    <parentprocess>

      <operator type="OR">

        <condition component="FileItem" property="FileName" condition="is" value="ex
cel.exe"/>

        <condition component="FileItem" property="FileName" condition="is" value="ms
access.exe"/>

        <condition component="FileItem" property="FileName" condition="is" value="ou
tlook.exe"/>

        <condition component="FileItem" property="FileName" condition="is" value="po
werpnt.exe"/>

        <condition component="FileItem" property="FileName" condition="is" value="wi
nword.exe"/>

      </operator>

    </parentprocess>

    <process>

      <operator type="OR">

        <condition component="FileItem" property="FileName" condition="is" value="po
wershell.exe"/>

        <condition component="FileItem" property="FileName" condition="is" value="cs
```

```
cript.exe"/>

        <condition component="FileItem" property="FileName" condition="is" value="ws
cript.exe"/>

        <condition component="FileItem" property="FileName" condition="is" value="cm
d.exe"/>

        <condition component="FileItem" property="FileName" condition="is" value="ms
hta.exe"/>

        <condition component="Module" property="OriginalFileName" condition="is" val
ue="powershell.exe"/>

        <condition component="Module" property="OriginalFileName" condition="is" val
ue="cscript.exe"/>

        <condition component="Module" property="OriginalFileName" condition="is" val
ue="wscript.exe"/>

        <condition component="Module" property="OriginalFileName" condition="is" val
ue="cmd.exe"/>

        <condition component="Module" property="OriginalFileName" condition="is" val
ue="mshta.exe"/>

      </operator>

    </process>

  </definition>

  <maliciousTarget name="current"/>

  <actions>

    <action name="TriggerDetection"/>

    <action name="StoreEvent"/>

  </actions>

</rule>
```

Things to notice in the rule example above:

1.We used process to identify the execution of the script interpreter and parentprocess to identify Microsoft Office application, so "Process executed by" is modeled by the process – parentprocess relation.

Working with LiveGrid and Safe property

This rule monitors suspicious executable modules dropped from rundll32.exe. Rundll32 is a Microsoft Windows system utility that provides an entry point and minimal framework for executing dynamic load libraries.

Rule

```xml
<?xml version='1.0' encoding='UTF-8'?>

<rule>

  <description>

    <name>Rundll32 Dropped Suspicious Executable [A0310]</name>

    <guid>d6359e46-f318-403c-b2b5-7133dd0fd0dd</guid>

    <category>File system</category>

    <os>Windows</os>

    <severity>61</severity>

    <mitreattackid>T1218.011,T1105</mitreattackid>

    <explanation>Rundll32 is a Microsoft Windows system utility that provides an entry point and minimal framework for executing dynamic load libraries. The rule monitors suspicious executable modules dropped from rundll32.exe</explanation>

    <benignCauses>May be part of some installation process.</benignCauses>

    <maliciousCauses>Rundll32 is commonly misused by malware</maliciousCauses>

    <recommendedActions>1. Evaluate the dropped module metadata.

2. Evaluate the executable drop reason from rundll32.exe.

3. Evaluate the rundll32.exe command line and loaded modules.

4. Evaluate the parent process, its command line and execution chain.</recommendedActions>

  </description>

  <definition>

    <process>

      <operator type="OR">

        <condition component="FileItem" property="FileName" condition="is" value="rundll32.exe"/>

        <condition component="Module" property="OriginalFileName" condition="is" value="RUNDLL32.exe"/>

      </operator>

    </process>

    <operations>

      <operation type="ModuleDrop">
```

```
        <operator type="AND">

          <condition component="LiveGrid" property="Popularity" condition="less" val
ue="1000"/>

          <condition component="LiveGrid" property="Reputation" condition="less" val
ue="8"/>

          <operator type="NOT">

            <operator type="OR">

              <condition component="Module" property="SignatureType" condition="is"
value="Trusted"/>

              <condition component="Enterprise" property="Safe" condition="is" value
="1"/>

            </operator>

          </operator>

        </operator>

      </operation>

    </operations>

  </definition>

  <maliciousTarget name="module"/>

  <actions>

    <action name="TriggerDetection"/>

    <action name="StoreEvent"/>

  </actions>

</rule>
```

Things to notice in the rule example above:

1. We used the operation ModuleDrop to detect dropped modules from rundll32.

2. To filter out most of the unwanted alerts for legitimate actions, we used three types of conditions:

   a. The Popularity and Reputation properties of LiveGrid component to eliminate widely used programs and programs with good reputation.

   b. The SignatureType property of Module component is Trusted.

   c. The Safe property of Enterprise component is set as True. You can explicitly mark a file as safe in ESET Inspect Web Console.

Working with compromised flag

Sometimes, adversary can inject malicious code into a legitimate running process. Unfortunately, similar code injection techniques are also used by a lot of legitimate software, e.g. screen readers for the visually impaired.

Creating detections for every CodeInjection event would generate too many false positives. To solve this issue, we can use Compromised flag in ESET Inspect.

Rule

First, we create a rule that contains the action MarkAsCompromised without the TriggerDetection action. The MarkAsCompromised will add a flag to the process that is on the receiving end of code injection.

```
<?xml version='1.0' encoding='UTF-8'?>

<rule>

  <description>

    <name>Common Injection Targets</name>

    <category>Special</category>

    <os>Windows</os>

    <severity>90</severity>

  </description>

  <definition>

    <operations>

      <operation type="CodeInjection">

        <operator type="AND">

          <condition component="CodeInjectionType" condition="is" property="CodeInje
ctionType" value="SetThreadContext"/>

          <operator type="OR">

            <condition component="FileItem" property="FileName" condition="is" value
="msedge.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="ComSvcConfig.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="explorer.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="DevicePairingWizard.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="EhStorAuthn.exe"/>
```

```
            <condition component="FileItem" property="FileName" condition="is" value
="Locator.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="WUAUCLT.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="WWAHost.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="WerFault.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="bootcfg.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="conhost.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="dllhost.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="getmac.exe"/>

            <condition component="FileItem" property="FileName" condition="is" value
="systray.exe"/>

          </operator>

        </operator>

      </operation>

    </operations>

  </definition>

  <maliciousTarget name="none"/>

  <actions>

    <action name="StoreEvent"/>

    <action name="MarkAsCompromised"/>

  </actions>

</rule>
```

Now with the compromised flag set, we can reference it in another rule when additional suspicious operation, such as accessing the LSASS process will occur.

```
<?xml version='1.0' encoding='UTF-8'?>

<rule>

  <description>
```

```
    <name>Credential Dumping From Compromised Process</name>

    <category>Suspicious process creation and process manipulation</category>

    <os>Windows</os>

    <severity>90</severity>

    <mitreattackid>T1003.001</mitreattackid>

    <explanation>A process has accessed the LSASS process in a way that is typical f
or Mimikatz. LSASS contains sensitive information such as credentials.</explanation>

    <benignCauses>Legitimate applications that access other running processes in an
improper way (e.g., certain installers).</benignCauses>

    <maliciousCauses>Adversary may access LSASS process in order to retrieve credent
ials - passwords and hashes.</maliciousCauses>

    <recommendedActions>1. Initiate Incident Response procedure.</recommendedActions
>

  </description>

  <definition>

    <process>

      <condition component="ProcessInfo" condition="is" property="Compromised" value
="1"/>

    </process>

    <operations>

      <operation type="OpenProcess">

        <operator type="AND">

          <condition component="FileItem" property="FileName" condition="is" value="
lsass.exe"/>

          <condition component="FileItem" property="Path" condition="is" value="%SYS
TEM%"/>

          <condition component="OpenProcess" property="AccessRight" condition="is" v
alue="4112"/>

        </operator>

      </operation>

    </operations>

  </definition>

  <maliciousTarget name="current"/>

  <actions>
```

```
    <action name="TriggerDetection"/>

    <action name="StoreEvent"/>

  </actions>

</rule>
```