

OSSIM

Correlation engine explained.

Sample scenario:
NETBIOS DCERPC ISystemActivator

Dominique Karg dk@ossim.net
<http://www.ossim.net>

Index

1. Foreword	3
2. Directives	4
3. NETBIOS DCERPC IsystemActivator	8
4. Conclusion	13
Appendix A – Plugins	14
Appendix B – Complete directive.....	15

1. Foreword

This short paper explains a sample directive used by ossim's correlation engine. It tries to show the pseudo-language used for directive composing, shows what each tag means and explains how and why a sample directive is built up.

This paper's main focus is logical directive correlation. OS / Service correlation and Snort / Nessus and it's implementations will be explained in future papers and will be included in a complete correlation paper that is about to be released.

For better understanding I'll resume how events are being processed by ossim agents and servers.

All event processing and normalization is done by lightweight agents that are able to read logs from:

- Snort
- Spade
- Apache
- IIS
- Iptables
- FW-1
- Etc...

The agent is also able to query ntop on information about certain hosts or established sessions. Opennms could also be considered a monitor since it can tell us if a certain service is running or not.

We differentiate between „monitor“ and „detector“ input. Detector input is continuously fed from the agents to the server as events or logs arrive.

Monitor input must be requested by the server, then the agents must query the monitors and report back to the server if the requested condition is met.

2. Logical Directives

OSSIM's directives are defined using xml 1.0. In order to show the possibilities we have as of today I'll try to explain the basic syntax.

The correlation engine reads all the directives on startup in order to match individual rules or events. It's functionality resembles a logical tree consisting of „if“ and „or“ statements, joined together to provide reliable means of identifying attacks or network misbehaviour.

A logical directive starts containing two tags. A directive id (decimal, unique) and a descriptive name that is shown when the directive is matched and it reaches an „alert“ status (more on this later).

```
<directive id="1" name="Successful Dcom exploit" priority="5">
```

Each directive has a starting rule that puts that begins to match the directive and starts the event correlation.

```
<rule type="detector" name="Snort dcom signature"  
  reliability="1" time_out="60" occurrence="1" from="ANY"  
  to="ANY" port_from="ANY" port_to="135,445" plugin_id="1001"  
  plugin_sid="2192">
```

This is our starting rule, i'll explain the fields step-by-step.

- Type

What type of rule is this. There are two possible types as of today:

1. Detector

Detector rules are those received automatically from the agent as they are recorded. This includes snort, spade, apache, etc...

2. Monitor

Monitor rules must be queried by the server ntop data and ntop sessions.

- Name

The rule name shown within the event database when the level is matched.

- Priority

The priority used within this directive. The only rule that should have a priority is the first one which, combined with the host asset and policy information will further qualify this attack.

One thing to note.

Snort uses inverse priority, 1 is high, 3 is low. Our approach is different. 0 means ignore, 1 means low and the higher the number the higher the priority.

- Reliability

Each rule has it's own reliability. This is an important variable as our correlation directive will usually be focused on „comprobatation“, this means that the first rule of a directive has the objective of locating a possible attack, and the rest of rules's only porpuse is to check wether this attack is really happening or it's just a false positive.

During a correlation process the reliability, which usually starts as „0“ or jut „1“ will grow as the correlation engine finds evidence that the attack is real. Reliability has a value between „0“ and „10“, and has the idea of real time probabibility measuared as a tenth of the percentage value: value „1“ will mean we think at this stage the event has 10% probability that the event is not a false positive, 3 will mean 30%, and so on.

- Time_out

We wait a fixed amount of seconds until a rule expires and the directives lifetime is over. We define the time-frame in which an attack may happen.

- Occurrence

How many times must a rule match until we advance to the next one? Combined with a high timeout this can be used to detect slow scan for example (using spade anomaly detector).

- From

Source IP. There are various possible values for this field:

1. ANY

Just that, any ip address would match.

2. Dotted numerical Ipv4 (x.x.x.x)

Self explaining.

3. Referenced.

This is used to reference ip addresses from previous levels. This should be easier to understand using examples:

1:SRC_IP means use the source ip referenced within the previous rule

2:DST_IP means use the destination ip referenced two rules below as source address

- To

Source IP. There are various possible values for this field:

1. ANY

Just that, any ip address would match.

2. Dotted numerical Ipv4 (x.x.x.x)

Self explaining.

3. Referenced.

This is used to reference ip addresses from previous levels. This should be easier to understand using examples

1:SRC_IP means use the source ip referenced within the previous rule

2:DST_IP means use the destination ip referenced two rules below as source address

The „To“ field is the field used when referencing monitor data that has no source.

Both „From“ and „To“ fields should accept input from the database in the near future. Host and Network objects are on the TODO list.

- Port_from

This can be a port number or a sequence of comma separated port numbers. ANY port can also be used. As before, source ports can also be referenced as in „1:SRC_PORT“ or „2:DST_PORT“.

- Port_to

This can be a port number or a sequence of comma separated port numbers. ANY port can also be used. As before, source ports can also be referenced as in „1:SRC_PORT“ or „2:DST_PORT“.

- Plugin_id

The numerical id assigned to the referenced plugin. See appendix A.

- Plugin_sid

The numerical sub-id assigned to each plugins events, functions or the like. For example, plugin id 1001 (snort) references it's rules as normal plugin_sids. Plugin id 1501 (apache) uses the response codes as plugin_sid (200 OK, 404 NOT FOUND, ...)

- Condition

This parameter and the following three are only valid for „monitor“ and certain „detector“ type rules.

The logical condition that has to be met for the rule to match:

1. eq - Equal
2. ne – Not equal
3. lt – Lesser than
4. gt – Greater than
5. le – Lesser or equal
6. ge – Greater or equal

- Value
-

The value that has to be matched using the previous directives.

- Interval
-

This value is similar to time_out but used for „monitor“ type rules.

- Absolute
-

Determines if the provided value is absolute or relative.

For example, providing 1000 as a value, gt as condition and 60 (seconds) as interval, querying ntop for HttpSentBytes would mean:

- Absolute true: Match if the host has more than 1000 http sent bytes within the next 60 seconds. Report back when (and only if) this absolute value is reached.
- Absolute false: Match if the host shows an increase of 1000 http sent bytes within the next 60 seconds. Report back as soon as this difference is reached (if it was reached...)

3. NETBIOS DCERPC IsystemActivator

After having learned the basic directive syntax let's examine our sample directive. I wrote this directive for testing purposes and some steps or rules could be wrong. Any feedback would be appreciated.

First, we have the start tag. This is the first directive so it has id=1 and it's name is „Successful Dcom exploit“ because that's what we'll try to catch.

```
<directive id="1" name="Successful Dcom exploit" priority="5">
```

Next we have our starting rule. This rule expects a snort alert number 2192 coming from any:any to any:135 or any:445. We only need one event to match this rule. When this event arrives we'll wait 60 seconds until a second level rule matches.

```
<rule type="detector" name="Snort dcom signature"
  reliability="1" time_out="60" occurrence="1" from="ANY"
  to="ANY" port_from="ANY" port_to="135,445" plugin_id="1001"
  plugin_sid="2192">
```

The next is a opening tag that means, after the previous rule match any of the next level rules.

```
<rules>
```

At second level our first rule tries to match a strange connection event from spade (plugin 1104, plugin_sid 1) using source and dest host data from the previous rule. Besides this, we raise the reliability to 3 and wait another 60 seconds for the next level to be reached. Again we only need this event once to match.

```
<rule type="detector" name="Strange open connection after 135/tcp or 445/tcp"
  reliability="3" time_out="60" occurrence="1" from="1:SRC_IP"
  to="1:DST_IP" port_from="ANY" port_to="ANY" plugin_id="1104"
  plugin_sid="1">
```

Next, another start tag and we advance to level three (if the two previous levels matched).

```
<rules>
```

This rule matches an attack response coming through the previous strange connection. This raises our reliability even more. Note that now we expect the event from the attacked host so we exchange source and dest host information. Ports must also be inverted but we didn't know them at level one so we use the information gleaned from level two.

```
<rule type="detector" name="Windows cmd.exe detected" reliability="6"
time_out="60" occurrence="1" from="1:DST_IP"
to="1:SRC_IP" port_from="2:DST_PORT" port_to="2:SRC_PORT"
plugin_id="1001" plugin_sid="2123">
```

Start one last level.

```
<rules>
```

Ok, what do we've got ? A dcom attack signature, a strange connection after that, and finally an attack response. Wow, that looks suspicious but let's see if we can glean a little bit more information.

What if the second, strange connection, has opened a persistent socket ? Let's query our monitor and see if that combination SRC_IP:source_port -> DST_IP:Strange_dest_port opens a connection that lasts more than 10 seconds. If so, the reliability is very very high and after four matched levels our directive expires.

For informational purposes, plugin 2005 is ntop and plugin_sid means „session duration“.

```
<rule type="monitor" name="Established session against abnormal port"
reliability="10"
from="1:SRC_IP" to="1:DST_IP" port_from="2:SRC_PORT"
port_to="2:DST_PORT" plugin_id="2005"
plugin_sid="248" condition="ge" value="10" interval="20" absolute="true" />
```

If that fails, let's wait five minutes and see if spade detects some strange connections originating from the attacked hosts, connections to odd or closed ports or worse, towards non-live hosts.

Compromise behaviour would normally be much better detected by the „compromise“ risk monitor, but for the sake of simplicity of the example we will make it directly with spade.

```
dests"
<rule type="detector" name="Attacked host is misbehaving. Nonlive
reliability="8" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>

<rule type="detector" name="Attacked host is misbehaving."
reliability="7" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>

<rule type="detector" name="Attacked host is misbehaving."
reliability="7" time_out="300" occurrence="3" from="1:DST_IP"
```

```
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>
```

After this, our directive would have accomplished its purpose and four separate events have shown that our example target host has certainly been compromised.

The next level 2, 3 and 4 level rules are similar but less reliable.

After this explanation you should be able to follow the remaining event flow from the directive shown at appendix B.

- The second second-level rule does the same as the first but expects different originating events.
- The third second-level rule is the same again, we expect a connect-back shellcode.
- The fourth watches for attack-responses that don't trigger the anomaly engine.
- The fifth and latter rules aren't that reliable and try to catch odd behaviour.

Ok, this is our final logical tree. Different levels use different colors.

Green: level 1

Yellow: level 2

Orange: level 3

Red: level 4

```
<rule type="detector" name="Snort dcom signature"
  <rules>
    <rule type="detector" name="Strange connection after 135/tcp or 445/tcp"
      <rules>
        <rule type="detector" name="Windows cmd.exe detected"
          <rules>
            <rule type="monitor" name="Established session against abnormal
              <rule type="detector" name="Attacked host is misbehaving. Nonlive
                <rule type="detector" name="Attacked host is misbehaving."
                  <rule type="detector" name="Attacked host is misbehaving."
                </rules>
              </rule>
            <rule type="monitor" name="Established session against odd port"
              <rules>
                <rule type="detector" name="Attacked host is misbehaving. Nonlive
                  <rule type="detector" name="Attacked host is misbehaving."
                    <rule type="detector" name="Attacked host is misbehaving."
                  </rules>
                </rule>
            <rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
              <rule type="detector" name="Attacked host is misbehaving."
                <rule type="detector" name="Attacked host is misbehaving."
              </rules>
            </rule>
          <rule type="detector" name="Odd connection after 135/tcp or 445/tcp"
            <rules>
              <rule type="detector" name="Windows cmd.exe detected"
            </rules>
```

```

    <rule type="monitor" name="Established session against compromised
    <rule type="detector" name="Attacked host is misbehaving. Nonlive
    <rule type="detector" name="Attacked host is misbehaving."
    <rule type="detector" name="Attacked host is misbehaving."
  </rules>
</rule>
<rule type="monitor" name="Established session against odd port"
<rules>
  <rule type="detector" name="Attacked host is misbehaving. Nonlive
  <rule type="detector" name="Attacked host is misbehaving."
  <rule type="detector" name="Attacked host is misbehaving."
</rules>
</rule>
<rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
<rule type="detector" name="Attacked host is misbehaving."
<rule type="detector" name="Attacked host is misbehaving."
</rules>
</rule>
<rule type="detector" name="Connect back after shellcode detected"
  <rules>
    <rule type="detector" name="Windows cmd.exe detected"
      <rules>
        <rule type="monitor" name="Established session from compromised host"
        <rule type="detector" name="Attacked host is misbehaving. Nonlive
        <rule type="detector" name="Attacked host is misbehaving."
        <rule type="detector" name="Attacked host is misbehaving."
      </rules>
    </rule>
    <rule type="monitor" name="Established session against odd port"
      <rules>
        <rule type="detector" name="Attacked host is misbehaving. Nonlive
        <rule type="detector" name="Attacked host is misbehaving."
        <rule type="detector" name="Attacked host is misbehaving."
      </rules>
    </rule>
    <rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
    <rule type="detector" name="Attacked host is misbehaving."
    <rule type="detector" name="Attacked host is misbehaving."
  </rules>
</rule>
<rule type="detector" name="Windows cmd.exe detected"
  <rules>
    <rule type="monitor" name="Established session against attacked host"
    <rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
    <rule type="detector" name="Attacked host is misbehaving."
    <rule type="detector" name="Attacked host is misbehaving."
  </rules>
</rule>
<rule type="monitor" name="Newly established session"
  <rules>
    <rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
    <rule type="detector" name="Attacked host is misbehaving."
    <rule type="detector" name="Attacked host is misbehaving."
  </rules>
</rule>
<rule type="detector" name="Strange connection after 135/tcp or 445/tcp"
<rule type="detector" name="Strange connection after 135/tcp or 445/tcp"
<rule type="detector" name="Odd connection after 135/tcp or 445/tcp"
</rules>

```

</rule>

4. Conclusion

Hopefully this paper shows a little bit of the power and flexibility of one part of the ossim correlation engine.

There is much more to write about all this because nothing is said about what happens when a directive is matched, what happens when a directive times out, what happens exactly with the priority and reliability but don't worry, that will all be included in the final correlation paper.

The syntax is also being rewritten and we want to provide some standard functions that can be used with directives, such as windows attack response with odd behaviour, attack response with established session and so on. This should make the directives much shorter, maybe 1/4 from what they're now because 80% is repetitive.

Everything is work in progress and probably some things will change. We have lots of ideas for improvements, both in functionality as in ease of use but I think it's a good idea to show some early work in order to expect real world input.

So, any ideas, critics, flames, etc... are welcome and if you have questions just ask them.

Appendix A – Plugins

Here is a list of plugins that can be used within directives as of today.

- 1001 | Snort Rules
- 1002 | Snort Tagging
- 1100 | Portscan1
- 1101 | Minfrag
- 1102 | HTTP decode 1/2
- 1103 | First defragmenter
- 1104 | SPADE
- 1105 | Back Orifice
- 1106 | RPC Preprocessor
- 1107 | 2nd stream preprocessor
- 1108 | 3rd stream preprocessor
- 1109 | Telnet option decoder
- 1110 | Unicode decoder
- 1111 | Stream4 preprocessor
- 1112 | Arp Spoof detector
- 1113 | 2nd fragment preprocessor
- 1114 | NOP detector
- 1115 | ASN.1 Validator
- 1116 | Snort Internal Decoder
- 1117 | Portscan2
- 1118 | Conversation
- 1119 | TBA
- 1120 | TBA
- 1121 | SNMP decoder
- 1501 | Apache
- 1502 | ISS
- 1503 | Iptables
- 1504 | FW1
- 1505 | OSSIM Directives alarms
- 2001 | OS-SIM
- 2002 | Arpwatch
- 2003 | POf
- 2004 | OpenNMS
- 2005 | NTop
- 3001 | Nessus
- 3002 | NMap

Appendix B – Complete directive

And finally, here is the finished sample directive. You can also download it from <http://www.ossim.net>

```
<directive id="1" name="Successful Dcom exploit" priority="5">

<!-- Originating rule. Watch for dcom snort signatures -->
<!-- This rule could also be rule 2190, 2191, 2193, etc... -->
  <rule type="detector" name="Snort dcom signature"
    reliability="1" time_out="60" occurrence="1" from="ANY"
    to="ANY" port_from="ANY" port_to="135,445" plugin_id="1001"
    plugin_sid="2192">

    <!-- Next, watch for a few things that could mean the intrusion was successful -->
    <rules>

      <!-- First watch for closed or rare open dest port.
      That could mean the attacker is connecting to a bindshell -->
      <rule type="detector" name="Strange connection after 135/tcp or 445/tcp"
        reliability="3" time_out="60" occurrence="1" from="1:SRC_IP"
        to="1:DST_IP" port_from="ANY" port_to="ANY" plugin_id="1104"
        plugin_sid="1">

    <rules>

      <!-- Now let's watch for an attack response. High reliability -->
      <rule type="detector" name="Windows cmd.exe detected" reliability="6"
        time_out="60" occurrence="1" from="1:DST_IP"
        to="1:SRC_IP" port_from="2:DST_PORT" port_to="2:SRC_PORT"
        plugin_id="1001" plugin_sid="2123">

    <rules>

      <!-- That's enough. Attack, odd port, attack response and
      established session. -->
      <rule type="monitor" name="Established session against compromised
      host" reliability="10"
        from="1:SRC_IP" to="1:DST_IP" port_from="2:SRC_PORT"
        port_to="2:DST_PORT" plugin_id="2005"
        plugin_sid="248" condition="ge" value="10" interval="20"
        absolute="true" />

      <!-- We've got an attack, odd dport, attack response and
      now the attacked host is having a strange behaviour. You'd
      better take a look -->
      <rule type="detector" name="Attacked host is misbehaving. Nonlive
      dests"
```

```
reliability="8" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>
```

```
<rule type="detector" name="Attacked host is misbehaving."
reliability="7" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>
```

```
<rule type="detector" name="Attacked host is misbehaving."
reliability="7" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>
```

```
</rules>
```

```
</rule>
```

```
<!-- Has there been a lasting session established ? Against an
odd port after RPC attack ? No good. But we missed the attack
response, let's see if the destination host shows odd
behaviour -->
```

```
<rule type="monitor" name="Established session against odd port"
reliability="5"
from="2:DST_IP" to="2:SRC_IP" port_from="2:SRC_PORT"
port_to="2:DST_PORT" plugin_id="2005"
plugin_sid="248" condition="ge" value="1" interval="20" absolute="true">
```

```
<rules>
```

```
<!-- Strange behaviour, increased reliability -->
```

```
<rule type="detector" name="Attacked host is misbehaving. Nonlive
dests"
```

```
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>
```

```
<rule type="detector" name="Attacked host is misbehaving."
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>
```

```
<rule type="detector" name="Attacked host is misbehaving."
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>
```

```
</rules>
```

```
</rule>
```

```
<!-- RPC attack detected, strange connection to host and host  
misbehaving ? bad. But we got no established session nor  
attack response so we can't be 100% sure. -->
```

```
<rule type="detector" name="Attacked host is misbehaving. Nonlive dests"  
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"  
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="3"/>
```

```
<rule type="detector" name="Attacked host is misbehaving."  
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"  
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="1"/>
```

```
<rule type="detector" name="Attacked host is misbehaving."  
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"  
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="6"/>
```

```
</rules>
```

```
</rule>
```

```
<!-- The source host could be local and being watched already so this  
could mean "source used odd dest port".  
Furthermore that could mean the attacker is connecting to a bindshell.  
Everything else same as previous rule. -->
```

```
<rule type="detector" name="Odd connection after 135/tcp or 445/tcp"  
reliability="3" time_out="60" occurrence="1" from="1:SRC_IP"  
to="1:DST_IP" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="6">
```

```
<rules>
```

```
<!-- Now let's watch for an attack response. High reliability -->
```

```
<rule type="detector" name="Windows cmd.exe detected" reliability="6"  
time_out="60" occurrence="1" from="1:DST_IP"  
to="1:SRC_IP" port_from="2:DST_PORT" port_to="2:SRC_PORT"  
plugin_id="1001" plugin_sid="2123">
```

```
<rules>
```

```
<!-- That's enough. Attack, odd port, attack response and  
established session. -->
```

```
<rule type="monitor" name="Established session against compromised  
host" reliability="10"
```

```

    from="1:SRC_IP" to="1:DST_IP" port_from="2:SRC_PORT"
port_to="2:DST_PORT" plugin_id="2005"
    plugin_sid="248" condition="ge" value="10" interval="20"
absolute="true" />

    <!-- We've got an attack, odd dport, attack response and
now the attacked host is having a strange behaviour. You'd
better take a look -->
    <rule type="detector" name="Attacked host is misbehaving. Nonlive
dests"
reliability="8" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>

    <rule type="detector" name="Attacked host is misbehaving."
reliability="7" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>

    <rule type="detector" name="Attacked host is misbehaving."
reliability="7" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>

</rules>

</rule>

    <!-- Has there been a lasting session established ? Against an
odd port after RPC attack ? No good. But we missed the attack
response, let's see if the destination host shows odd
behaviour -->
    <rule type="monitor" name="Established session against odd port"
reliability="5"
    from="2:DST_IP" to="2:SRC_IP" port_from="2:SRC_PORT"
port_to="2:DST_PORT" plugin_id="2005"
    plugin_sid="248" condition="ge" value="1" interval="20" absolute="true">

<rules>

    <!-- Strange behaviour, increased reliability -->
    <rule type="detector" name="Attacked host is misbehaving. Nonlive
dests"
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>

    <rule type="detector" name="Attacked host is misbehaving."
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"

```

```
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>

<rule type="detector" name="Attacked host is misbehaving."
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>

</rules>

</rule>

<!-- RPC attack detected, strange connection to host and host
misbehaving ? bad. But we got no established session nor
attack response so we can't be 100% sure. -->
<rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>

<rule type="detector" name="Attacked host is misbehaving."
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>

<rule type="detector" name="Attacked host is misbehaving."
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>

</rules>

</rule>

<!-- If the attacked host connects back right after the attack is
being detected we start to get worried. Some port references change
but everything else is the same as previous rules -->
<rule type="detector" name="Connect back after shellcode detected"
reliability="3" time_out="60" occurrence="1" from="1:DST_IP"
to="1:SRC_IP" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1">

<rules>

<!-- Now let's watch for an attack response. High reliability -->
<rule type="detector" name="Windows cmd.exe detected" reliability="6"
time_out="60" occurrence="1" from="1:DST_IP"
to="1:SRC_IP" port_from="2:SRC_PORT" port_to="2:DST_PORT"
plugin_id="1001" plugin_sid="2123">
```

```

<rules>
    <!-- That's enough. Attack, connect-back, attack response and
    established session. -->
    <rule type="monitor" name="Established session from compromised host"
reliability="10"
    from="1:DST_IP" to="1:SRC_IP" port_from="2:SRC_PORT"
port_to="2:DST_PORT" plugin_id="2005"
    plugin_sid="248" condition="ge" value="10" interval="20"
absolute="true" />

    <!-- We've got an attack, connect-back, attack response and
    now the attacked host is having a strange behaviour. You'd
    better take a look -->
    <rule type="detector" name="Attacked host is misbehaving. Nonlive
dests"
reliability="8" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>

    <rule type="detector" name="Attacked host is misbehaving."
reliability="7" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>

    <rule type="detector" name="Attacked host is misbehaving."
reliability="7" time_out="300" occurrence="3" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>

</rules>

</rule>

    <!-- Has there been a lasting session established ? With a
    connection back to the attacking host after a RPC attack ?
    No good. But we missed the attack response, let's see if the
    destination host shows odd behaviour -->
    <rule type="monitor" name="Established session against odd port"
reliability="5"
    from="1:DST_IP" to="1:SRC_IP" port_from="2:SRC_PORT"
port_to="2:DST_PORT" plugin_id="2005"
    plugin_sid="248" condition="ge" value="1" interval="20" absolute="true">

<rules>

    <!-- Strange behaviour, increased reliability -->

```

```
dests"
    <rule type="detector" name="Attacked host is misbehaving. Nonlive
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>

    <rule type="detector" name="Attacked host is misbehaving."
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>

    <rule type="detector" name="Attacked host is misbehaving."
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>

</rules>

</rule>

<!-- RPC attack detected, strange connection to host and host
misbehaving ? bad. But we got no established session nor
attack response so we can't be 100% sure. -->
<rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="3"/>

<rule type="detector" name="Attacked host is misbehaving."
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="1"/>

<rule type="detector" name="Attacked host is misbehaving."
reliability="4" time_out="300" occurrence="10" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
plugin_sid="6"/>

</rules>

</rule>

<!-- Ok, the exploit could have killed the process and returned the
shell so let's see if we catch a shell string. No anomalies so far,
let's wait some more. -->
<rule type="detector" name="Windows cmd.exe detected" reliability="5"
time_out="60" occurrence="1" from="1:DST_IP"
```

```

    to="1:SRC_IP" port_from="1:DST_PORT" port_to="1:SRC_PORT"
    plugin_id="1001" plugin_sid="2123">

    <rules>

        <!-- Attack, attack response and established session. No good. -->
        <rule type="monitor" name="Established session against attacked host"
reliability="8"
    from="1:DST_IP" to="1:SRC_IP" port_from="1:SRC_PORT"
port_to="1:DST_PORT" plugin_id="2005"
    plugin_sid="248" condition="ge" value="10" interval="20" absolute="true"
/>

        <!-- We've got an attack, attack response and now the attacked host
is having a strange behaviour. You'd better take a look -->
        <rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
    plugin_sid="3"/>

        <rule type="detector" name="Attacked host is misbehaving."
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
    plugin_sid="1"/>

        <rule type="detector" name="Attacked host is misbehaving."
reliability="6" time_out="300" occurrence="5" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"
    plugin_sid="6"/>

    </rules>

</rule>

    <!-- Let's face it, maybe we didn't catch anything using anomalies or
signatures. But... could there be a recent connection originating from
the target host to the attacker ? -->
    <rule type="monitor" name="Newly established session" reliability="4"
from="1:DST_IP"
to="1:SRC_IP" port_from="ANY" port_to="ANY" plugin_id="2005"
    plugin_sid="248" condition="le" value="60" interval="20" absolute="true">

    <rules>

        <!-- Yeah, there is a session. Does the target host misbehave ? -->
        <rule type="detector" name="Attacked host is misbehaving. Nonlive dests"
reliability="5" time_out="300" occurrence="8" from="1:DST_IP"
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"

```

```
plugin_sid="3"/>
```

```
<rule type="detector" name="Attacked host is misbehaving."  
reliability="5" time_out="300" occurrence="8" from="1:DST_IP"  
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="1"/>
```

```
<rule type="detector" name="Attacked host is misbehaving."  
reliability="5" time_out="300" occurrence="8" from="1:DST_IP"  
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="6"/>
```

```
</rules>
```

```
</rule>
```

```
<!-- One last try. Maybe no shell is being spawned, nor does the  
attacker initiate any connections nor does the host connect back. So,  
what if the host starts generating odd traffic...
```

```
This doesn't mean much so the reliability isn't high and we need a  
couple of strange connections until we have a match but we'll have  
some patience. -->
```

```
<rule type="detector" name="Strange connection after 135/tcp or  
445/tcp to non live hosts"  
reliability="2" time_out="300" occurrence="10" from="1:DST_IP"  
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="3"/>
```

```
<rule type="detector" name="Strange connection after 135/tcp or 445/tcp"  
reliability="2" time_out="300" occurrence="10" from="1:DST_IP"  
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="1"/>
```

```
<rule type="detector" name="Odd connection after 135/tcp or 445/tcp"  
reliability="2" time_out="300" occurrence="10" from="1:DST_IP"  
to="ANY" port_from="ANY" port_to="ANY" plugin_id="1104"  
plugin_sid="6"/>
```

```
<!-- Note: this rule could catch mass mailing worms, i.e. the host  
normally sends mail using server A and now uses server B,C and D. -->
```

```
</rules>
```

```
</rule>
```

```
</directive>
```

```
</directives>
```