Nessus File (.nessus) Format

Last Revised: May 24, 2023

Table of Contents

Overview

This guide describes the Tenable Nessus file (`.nessus`) format structure.

You can use the `.nessus` file format for importing and exporting scans in Tenable Nessus. The format has the following advantages:

- XML based – Forward and backward compatibility and easy implementation.
- Self-sufficient – A single `.nessus` file contains the list of targets, the policies defined by the user, and the scan results.
- Secure – Tenable Nessus does not save passwords in the file. Instead, Tenable Nessus stores a reference to a password, which is stored in a secure location on the local host.

File Structure

The `.nessus` file format lists two sections: `Policy` and `Report`. Each section can have multiple components. View the following example, which includes a **NessusClientData** header and footer:

```
<NessusClientData>
<Policy><policyName>My Policy</policyName> [..]
</Policy>
<Report name="My Scan"> [..]
</Report>
</NessusClientData>
```

Note: A single `.nessus` file might only contain a policy or a scan policy with reported results.

Policies

The `Policy` section is the most detailed section of the `.nessus` file format. This section enables and disables individual plugins and plugin families, sets individual plugin preferences, and specifies credentials. It also allows for a unique name and description. The following example demonstrates the `Policy` section structure. *policyName* is the name of the policy, and *policyComments* is the comment associated to this policy.

```
<Policy>
<policyName>My Name</policyName>
<policyComments>My Comment</policyComments>
<Preferences>
<ServerPreferences>
      <preference>
            <name>max_simult_tcp_sessions</name>
            <value>unlimited</value>
      </preference>
</ServerPreferences>
<PluginsPreferences>
      <item>
            <pluginName>Web Application Tests Settings</pluginName>
            <pluginId>39471</pluginId>
            <fullName>Web Application Tests Settings[checkbox]:Enable web applications
tests</fullName>
            <preferenceName>Enable web applications tests</preferenceName>
            <preferenceType>checkbox</preferenceType>
            <preferenceValues>no</preferenceValues>
            <selectedValue>no</selectedValue>
      </item>
</PluginsPreferences>
</Preferences>
<FamilySelection>
      <FamilyItem>
            <FamilyName>MacOS X Local Security Checks</FamilyName>
            <Status>disabled</Status>
      </FamilyItem>
</FamilySelection>
</Policy>
```

Preference Component

The `Preferences` component within the `Policy` component contains two elements: `ServerPreferences` and `PluginPreferences`.

`ServerPreferences` element

The `ServerPreferences` element specifies configuration parameters for the remote Nessus scanner, such as values `max_host`, `port_range`, `unscanned_closed`. The sub-items for the `ServerPreferences` element are named `preference`. Each `preference` indicates a preference name and value. See the following example:

```
<preference>
      <name>[prefName]</name>
      <value>[prefValue]</value>
</preference>
```

Here is an example `ServerPreferences` element with multiple `preference` sections:

```
<ServerPreferences>
      <preference>
            <name>max_hosts</name>
            <value>10</value>
      </preference>
      <preference>
            <name>max_checks</name>
            <value>3</value>
      </preference>
</ServerPreferences>
```

`PluginsPreferences` element

The `PluginsPreferences` element specifies the configuration parameters for the plugins within a scan policy. This element includes an `item` for each Nessus plugin preference. Its structure is slightly more detailed than the `ServerPreferences` component because it includes the raw plugin preference text returned from the Nessus scanner and pre-processed values, both of which allow `.nessus` files to load faster. A `PluginsPreferences` element does not *need* any `item` sections. See the following example of an *item*:

```
<item>
      <pluginName>[theNameThePreferenceIsAttachedTo]</pluginName>
      <pluginId>[Plugin ID Number]</pluginId>
      <fullName>[PreferenceNameAsSentByNessusd</fullName>
      <preferenceName>[Parsed Name]</preferenceName>
      <preferenceType>[entry|radio|checkbox|file|password]</preferenceType>
      <preferenceValues>[the values as sent by nessusd]</preferenceValues>
      <selectedValue>[value selected by the end user]</selectedValue>
</item>
```

For each preference, the `fullName` variable contains all of the data necessary to derive the `preferenceName`, `preferenceType`, and `pluginName` content.

If the `preferenceType` variable is set to `password`, then it is not saved on disk (it would be considered a security vulnerability), unless the `Policy` has had an attribute of `passwordsType` set to **Clear Text**. Instead, Tenable Nessus uses a UUID that designates the password, which is saved in the local host's secure storage (for example, KeyChain on macOS). See the following example:

```
<PluginsPreferences>
      <item>
            <pluginName>Ping the remote host</pluginName>
            <pluginId>10180</pluginId>
            <fullName>Ping the remote host[entry]:TCP ping destination ports:</-
fullName>
            <preferenceName>TCP ping destination ports:</preferenceName>
            <preferenceType>entry</preferenceType>
            <preferenceValues>built-in</preferenceValues>
            <selectedValue>built-in</selectedValue>
      </item>
      <item>
            <pluginName>Ping the remote host</pluginName>
            <pluginId>10180</pluginId>
            <fullName>Ping the remote host[checkbox]:Do an ARP ping</fullName>
            <preferenceName>Do an ARP ping</preferenceName>
            <preferenceType>checkbox</preferenceType>
            <preferenceValues>yes</preferenceValues>
            <selectedValue>yes</selectedValue>
      </item>
      <item>
            <pluginName>Ping the remote host</pluginName>
```

```
            <pluginId>10180</pluginId>
            <fullName>Ping the remote host[checkbox]:Do a TCP ping</fullName>
            <preferenceName>Do a TCP ping</preferenceName>
            <preferenceType>checkbox</preferenceType>
            <preferenceValues>yes</preferenceValues>
            <selectedValue>yes</selectedValue>
      </item>
      <item>
            <preferenceName>Test SSL-based services</preferenceName>
            <pluginId>22964</pluginId>
            <fullName>Services[radio]:Test SSL-based services</fullName>
            <pluginName>Services</pluginName>
            <preferenceType>radio</preferenceType>
            <preferenceValues>Known SSL ports;All;None</preferenceValues>
            <selectedValue>Known SSL ports</selectedValue>
      </item>
</PluginsPreferences>
```

Plugin Selection Component

The `PluginSelection` component within the `Policy` component contains two elements: `FamilySelection` and `InvidivualPluginSelection`. This component allows you to enable and disable Nessus families and individual plugins.

`FamilySelection` element

A plugin family can have the state of `enabled`, `disabled`, or `partial`.

If a family is enabled, then all plugins from within that family are enabled, even if they have recently been added to a Nessus scanner.

If a family is disabled, then all plugins from that family are not enabled. Although a plugin might not be enabled within a policy, if the plugin is a dependency of another plugin and the policy enables dependencies, this plugin may eventually be used in a scan.

Lastly, you can mark a family as partially enabled. This means that one or more plugins from within a family have been enabled, but other plugins are not enabled. In this case, the `PluginItem` section determines the plugin status. If a family is placed into partial mode, plugins are not enabled by default. This also means that, as a developer or scan policy creator, you can choose to include only the enabled plugins, which can minimize the size of your `.nessus` file considerably.

See the following example of the `FamilyItem` element within `FamilySelection`:

```
<FamilyItem>
      <FamilyName>[familyName]</FamilyName>
      <Status>[enabled|disabled|partial]</Status>
</FamilyItem>
```

See the following example of a populated `FamilyItem` element that enables the FTP plugin family:

```
<FamilyItem>
      <FamilyName>FTP</FamilyName>
      <Status>enabled</Status>
</FamilyItem>
```

`IndividualPluginSelection` element

The `IndividualPluginSelection` element itemizes which plugins have been enabled for families that have been placed into `partial` mode. This element is made up of zero or more of the following items:

```
<PluginItem>
      <PluginId>[PluginID]</PluginId>
      <PluginName>[PluginName]</PluginName>
      <Family>[PluginFamily]</Family>
      <Status>[enabled|disabled]</Status>
</PluginItem>
```

See the following example of a populated `PluginItem` for plugin 10796:

```
<PluginItem>
      <PluginId>10796</PluginId>
      <PluginName>scan for LaBrea tarpitted hosts</PluginName>
      <Family>Port scanners</Family>
      <Status>disabled</Status>
</PluginItem>
```

Populated Policies Example

See the following fully populated example of a `Policy` section:

```
<Policies>
<Policy>
      <policyName>My Example Policy</policyName>
      <policyComment>This is an example policy</policyComment>
      <Preferences>
            <ServerPreferences>
                  <preference>
                        <name>max_hosts</name>
                        <value>30</value>
                  </preference>
            </ServerPreferences>
            <PluginsPreferences>
                  <item>
                        <pluginName>Web Application Tests Settings</pluginName>
                        <pluginId>39471</pluginId>
                        <fullName>Web Application Tests Settings[checkbox]:Enable web
applications tests</fullName>
                        <preferenceName>Enable web applications tests</preferenceName>
                        <preferenceType>checkbox</preferenceType>
                        <preferenceValues>no</preferenceValues>
                        <selectedValue>no</selectedValue>
                  </item>
            </PluginsPreferences>
      </Preferences>
      <PluginSelection>
            <FamilySelection>
                  <FamilyItem>
                        <FamilyName>Web Servers</FamilyName>
                        <Status>disabled</Status>
                  </FamilyItem>
            </FamilySelection>
            <IndividualPluginSelection>
                  <PluginItem>
                        <PluginId>34220</PluginId>
                        <PluginName>netstat port scanner (WMI)</PluginName>
                        <Family>Port scanners</Family>
                        <Status>enabled</Status>
```

```
                </PluginItem>
            </IndividualPluginSelection>
      </PluginSelection>
</Policy>
</Policies>
```

Report Function

The `Report` section can contain zero or one report per `.nessus` file. It includes a specific report name, the target, and scan results.

The following is a template of how the `Report` section is formatted:

```
<Report name="Router - Uncredentialed">
<ReportHost name="192.168.0.1">
      <HostProperties>
            [..]
      </HostProperties>
      <ReportItem>
            [..]
      </ReportItem>
</ReportHost>
</Report>
```

Report Host Component

The `ReportHost` component within the `Report` section contains all the findings for each host and metadata such as the scan's start and stop times, MAC address, detected operating system, and a summary of vulnerabilities found by severity. The component lists one vulnerability per `ReportItem` directive and includes a vulnerability synopsis, description, solution, references, and relevant plugin output for each vulnerability.

The following is an example for the `ReportHost` component:

```
<ReportHost name="192.168.0.10">
<HostProperties>
      <tag name="HOST_END">Wed Mar 09 22:55:00 2011</tag>
      <tag name="operating-system">Microsoft Windows XP Professional (English)</tag>
      <tag name="mac-address">00:1e:8c:83:ad:5f</tag>
      <tag name="netbios-name">ZESTY</tag>
      <tag name="HOST_START">Wed Mar 09 22:48:10 2011</tag>
</HostProperties>
<ReportItem> [..] </ReportItem>
<ReportItem> [..] </ReportItem>
<ReportItem> [..] </ReportItem>
</ReportHost>
```

Various `<tag>` directives such as `operating-system` and `netbios-name` are optional, and only included when the data is available. When creating a `.nessus` file, you can add them, and a Nessus client parsing this report can compute this information from the data at hand.

ReportItem Element

The *ReportItem* element represents one finding on a given port and host. Its structure is outlined in the following example:

```
<ReportItem port="445" svc_name="cifs" protocol="tcp" severity="3" pluginID="49174"
pluginName="Opera &lt; 10.62 Path Subversion Arbitrary DLL Injection Code Execution"
pluginFamily="Windows">
      <exploitability_ease>Exploits are available</exploitability_ease>
      <vuln_publication_date>2010/08/24</vuln_publication_date>
      <cvss_temporal_vector>CVSS2#E:F/RL:W/RC:ND</cvss_temporal_vector>
      <solution>Upgrade to Opera 10.62 or later.</solution>
      <cvss_temporal_score>8.4</cvss_temporal_score>
      <risk_factor>High</risk_factor>
      <description>The version of Opera installed on the remote host is earlier than
10.62. Such versions insecurely look
      in their current working directory when resolving DLL dependencies, such as for
&apos;dwmapi.dll&apos; [..]
      </description>
      <plugin_publication_date>2010/09/10</plugin_publication_date>
      <cvss_vector>CVSS2#AV:N/AC:M/Au:N/C:C/I:C/A:C</cvss_vector>
      <synopsis>The remote host contains a browser that allows arbitrary code
execution.</synopsis>
      <patch_publication_date>2010/09/09</patch_publication_date>
      <see_also>http://www.opera.com/docs/changelogs/windows/1062/</see_also>
      <see_also>http://www.opera.com/support/kb/view/970/</see_also>
      <exploit_available>true</exploit_available>
      <plugin_modification_date>2010/12/23</plugin_modification_date>
      <cvss_base_score>9.3</cvss_base_score>
      <bid>42663</bid>
      <xref>OSVDB:67498</xref>
      <xref>Secunia:41083</xref>
      <xref>EDB-ID:14732</xref>
      <plugin_output>
      Path : C:\Program Files\Opera
      Installed version : 9.52
      Fixed version : 10.62
      </plugin_output>
      <plugin_version>$Revision: 1.3 $</plugin_version>
</ReportItem>
```

When you use a compliance check, the `ReportItem` element adds additional `cm` related tags to denote the compliance information:

```
<ReportItem port="0" svc_name="general" protocol="tcp" severity="3" pluginID="21157"
pluginName="Unix Compliance Checks" pluginFamily="Policy Compliance">
      <fname>unix_compliance_check.nbin</fname>
      <plugin_modification_date>2012/06/20</plugin_modification_date>
      <plugin_name>Unix Compliance Checks</plugin_name>
      <plugin_publication_date>2006/03/27</plugin_publication_date>
      <plugin_type>local</plugin_type>
      <risk_factor>None</risk_factor>
      <cm:compliance-info>Local and remote storage of Apache error logs is critical to
successful break-in investigation
      and should be configured via the syslog facility. ref. CIS_Apache_Benchmark_
v2.1.pdf, ch. 1, pp 44-46. Checking
      that your Apache configuration file contains the proper syslog
entry.</cm:compliance-info>
      <cm:compliance-result>FAILED</cm:compliance-result>
      <cm:compliance-actual-value>The file
&amp;quot;/usr/local/apache2/conf/httpd.conf&amp;quot; could not be
      found</cm:complianceactual-value>
      <cm:compliance-check-id>0380a6f83735bfd70235e8da91821049</cm:compliancecheck-id>
      <cm:compliance-audit-file>CIS_Apache_v2_1.audit</cm:compliance-audit-file>
      <cm:compliance-check-name>2.5 Syslog Logging. (ErrorLog)</cm:compliance-check-
name>
      <description>&amp;quot;2.5 Syslog Logging. (ErrorLog)&amp;quot; : [FAILED] Local and
remote storage of Apache error logs is
      critical to successful break-in investigation and should be configured via the
syslog facility. ref. CIS_Apache_
      Benchmark_v2.1.pdf, ch. 1, pp 44-46.Checking that your Apache configuration file
contains the proper syslog entry.
      - error message: The file &amp;quot;/usr/local/apache2/conf/httpd.conf&amp;quot; could
not be found </description>
</ReportItem>
```

The following table lists and describes `ReportItem` elements. Not all elements are included for every `ReportItem`.

Element	Description

Port	The port number for the associated finding.
svc_name	The service name, if known.
Protocol	The protocol used to communicate (for example, TCP or UDP).
Severity	The severity level: • 0 – informational • 1 – low • 2 – medium • 3 – high • 4 – critical
pluginID	The numeric ID of the plugin.
pluginName	The plugin title.
pluginFamily	The family the plugin belongs to.
risk_factor	The risk factor (Low, Medium, High, or Critical).
synopsis	A brief description of the plugin or vulnerability.
description	The full text description of the vulnerability.
solution	The remediation information for the vulnerability.
plugin_output	The text output of the Nessus scanner.
see_also	A list of external references to additional information regarding the vulnerability.
cve	The CVE reference.
bid	The Bugtraq ID (BID) reference.
xref	The external reference (for example, OSVDB, Secunia, MS Advisory).
plugin_modification_date	The date the Nessus plugin was last modified.

plugin_publication_date	The date the Nessus plugin was published.
patch_publication_date	The date a vulnerability patch was published.
vuln_publication_ date	The date the vulnerability was published.
exploitability_ ease	The description of how easy it is to exploit the issue.
exploit_available	Describes whether a public exploit exists for the vulnerability.
exploit_framework_can-vas	Describes whether an exploit exists in the Immunity CANVAS framework.
exploit_framework_ metasploit	Describes whether an exploit exists in the Metasploit framework.
exploit_framework_core	Describes whether an exploit exists in the CORE Impact frame-work.
metasploit_name	The name of the Metasploit exploit module.
canvas_package	The name of the CANVAS exploit package.
cvss_vector	The Common Vulnerability Scoring System (CVSSv2) score.
cvss_base_score	The CVSSv2 base score (intrinsic and fundamental char-acteristics of a vulnerability that are constant over time and user environments).
cvss_temporal_ score	The CVSSv2 temporal score (characteristics of a vulnerability that change over time but not among user environments).
plugin_type	The general type of plugin check (for example, local or remote).
plugin_version	The version of the Nessus plugin used to perform the check.
cm:complianceinfo	Detailed information about the compliance check performed.
cm:complianceresult	The result of the compliance check (for example, PASSED or FAILED).
cm:complianceactual-value	Relevant output related to the compliance check.

cm:compliancecheck-id	The ID for the compliance check performed.