tenable network security

Nessus v2 File Format

October 15, 2013

(Revision 3)

Table of Contents

Introduction

This document covers the Nessus file format structure for the version 2 **.nessus** file, which was introduced with Nessus 4.0. Please share your comments and suggestions with us by emailing them to support@tenable.com.

A basic understanding of the Nessus vulnerability scanner is assumed.

Standards and Conventions

Throughout the documentation, filenames, daemons and executables are indicated with a **courier bold** font such as **gunzip**, **httpd** and **/etc/passwd**.

Command line options and keywords are also indicated with the **courier bold** font. Command line options may or may not include the command line prompt and output text from the results of the command. Often, the command being run will be **boldfaced** to indicate what the user typed. Below is an example running of the Unix **pwd** command:

```
# pwd
/opt/nessus/
#
```

Important notes and considerations are highlighted with this symbol and grey text boxes.

Tips, examples, and best practices are highlighted with this symbol and white on blue text.

Overview

NessusClient 3.2 introduced a new file format (**.nessus**) for scan export and import. This format was enhanced and labeled version 2 with the release of Nessus 4.0. The format has the following advantages:

- XML based, for easy forward and backward compatibility and easy implementation.
- Self-sufficient: a single **.nessus** file contains the list of targets, the policies defined by the user as well as the scan results themselves.
- Secure: passwords are not saved in the file. Instead a reference to a password stored in a secure location on the local host is used.

With the release of Nessus 4.0, the **.nessus** file format was updated. The previous format is still supported, but has been designated as **.nessus** (v1). All references to the **.nessus** format in this document denote version 2, the default format used when **.nessus** is selected.

File Structure

The **.nessus** file format lists two sections named “Policy” and “Report”. Each section can have multiple components. A basic outline is shown below, including the “NessusClientData” header and footer:

```
<NessusClientData_v2>
<Policy>
```

```
<policyName>My Policy</policyName>
[..]
</Policy>
<Report name="My Scan">
[..]
</Report>
</NessusClientData_v2>
```

It is important to realize that a single **`.nessus`** file might only contain a policy or a scan policy with reported results.

Policies Section

The most sophisticated portion of the **`.nessus`** file format is the "**Policy**" section. This section enables and disables families, individual plugins, sets individual plugin preferences and specifies credentials. It also allows for a unique name and description. Below is the structure of a "**Policy**" section:

```
<Policy>
<policyName>My Name</policyName>
<policyComments>My Comment</policyComments>
<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>
```

- **policyName** is the name of the policy.
- **policyComments** is the comment associated to this policy.

Preferences Component

The "**Preferences**" component within the "**Policy**" component contains two elements: "**ServerPreferences**", "**PluginPreferences**" and "**FamilySelection**".

ServerPreferences Element

The "**ServerPreferences**" element is used to specify configuration parameters for the remote Nessus scanner. These typically include values for "**max_host**", "**port_range**", "**unscanned_closed**" and so on. The sub-items for the "**ServerPreferences**" element are named "**preference**". Each "preference" indicates a preference name and value such as:

```
<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

To specify the configuration parameters for the plugins within a scan policy, the "**PluginsPreferences**" element is used. This element includes an "**item**" for each Nessus plugin preference. Its structure is slightly more complex than the "ServerPreferences" component because it includes both the raw plugin preference text returned from the Nessus scanner as well as pre-processed values. This makes loading a `.nessus` file faster. A "**PluginsPreferences**" element does not need any "item" sections.

Below is an example template for 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 enduser]</selectedValue>
</item>
```

For each preference, the "**fullName**" variable will contain all of the data necessary to derive the "preferenceName", "preferenceType" and "pluginName" content. In addition, keep in mind that 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" as mentioned previously. Instead, a UUID designating it on the local host secure storage (KeyChain on Mac OS X, etc.) is used.

Below is an example:

```
<PluginsPreferences>
 <item>
  <pluginName>Ping the remote host</pluginName>
  <pluginId>10180</pluginId>
 <fullName>Ping the remote host[entry]:TCP ping destination port(s):</fullName>
 <preferenceName>TCP ping destination port(s):</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>
```

PluginSelection Component

The “**PluginSelection**” component within the “Policy” component contains two elements: “**FamilySelection**” and “**InvidivualPluginSelection**”. This component allows Nessus families to be completely enabled and disabled as well as individual plugins to be enabled and disabled.

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 will be enabled, even if they have recently been added to a Nessus scanner.

If a family is disabled, then all plugins from that family will not be enabled. Keep in mind that 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, a family can be marked as being 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 status of a plugin is determined by the “PluginItem”

section. If a family is placed into partial mode, plugins will not be enabled by default. This also means that as a developer or scan policy creator, you can choose to include only the enabled plugins, which will considerably minimize the size of your `.nessus` file.

Below is a template for the "**FamilyItem**" element within "**FamilySelection**":

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

Below is an example 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 specifically 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>
```

Here is an example 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

Below is a 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 portscanner (WMI)</PluginName>
      <Family>Port scanners</Family>
      <Status>enabled</Status>
     </PluginItem>
    </IndividualPluginSelection>
   </PluginSelection>
  </Policy>
</Policies>
```

Report Section

The “**Report**” section can contain zero or one report per **`.nessus`** file. It is organized by specific report name and includes the target and results of the scan.

Below 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>
```

ReportHost Component

The "**ReportHost**" component within the "**Report**" section contains all of the findings for each host including some metadata such as the start and stop time of the scan, MAC address, detected operating system and a summary of vulnerabilities found by severity. Vulnerabilities are listed one per "**ReportItem**" directive and include vulnerability synopsis, description, solution, references and relevant plugin output.

Below 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>
```

Note that various <tag> directives such as "**operating-system**" and "**netbios-name**" are optional, and only included when the data is available. A client creating a **`.nessus`** file may choose to add them and a Nessus client parsing this report should be able to compute this information from the data at hand.

ReportItem Element

The "**ReportItem**" element is one finding on a given port on a given host. Its structure is outlined in this 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 web 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 a compliance check is used, the "**ReportItem**" element will add 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 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 &quot;/usr/local/apache2/conf/httpd.conf&quot;
       could not be found</cm:compliance-actual-value>
<cm:compliance-check-id>0380a6f83735bfd70235e8da91821049</cm:compliance-check-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>&quot;2.5 Syslog Logging. (ErrorLog)&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 your Apache configuration file contains the proper syslog entry.
 - error message: The file &quot;/usr/local/apache2/conf/httpd.conf&quot; could not be
       found </description>
</ReportItem>
```

The table below lists the various "**ReportItem**" elements and gives a brief description of each. Not all elements will be included for every ReportItem.

The list of elements will grow and change over time, as more information is added to the Nessus plugins.

ReportItem Element	Description
Port	Port number for the associated finding
svc_name	The service name, if known

Protocol	The protocol used to communicate (e.g., TCP, UDP)
Severity	Severity level corresponds with a “0” as an open port, “1” as a low or informational, “2” as a medium or warning and “3” as a high or hole
pluginID	Numeric ID of the plugin
pluginName	Title of the plugin
pluginFamily	Family the plugin belongs to
risk_factor	Low, Medium, High or Critical risk factor
synopsis	Brief description of the plugin or vulnerability
description	Full text description of the vulnerability
solution	Remediation information for the vulnerability
plugin_output	Actual text output of the Nessus scanner
see_also	External references to additional information regarding the vulnerability
cve	CVE reference
bid	Bugtraq ID (BID) reference
xref	External reference (e.g., OSVDB, Secunia, MS Advisory)
plugin_modification_date	Date the Nessus plugin was last modified
plugin_publication_date	Date the Nessus plugin was published
patch_publication_date	Date a patch for the vulnerability was published
vuln_publication_date	Date the vulnerability was published
exploitability_ease	Description of how easy it is to exploit the issue
exploit_available	If a public exploit exists for the vulnerability
exploit_framework_canvas	If an exploit exists in the Immunity CANVAS framework

exploit_framework_metasploit	If an exploit exists in the Metasploit framework
exploit_framework_core	If an exploit exists in the CORE Impact framework
metasploit_name	The name of the Metasploit exploit module
canvas_package	The name of the CANVAS exploit package
cvss_vector	Common Vulnerability Scoring System (CVSS) version 2 score
cvss_base_score	CVSSv2 base score (intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments)
cvss_temporal_score	CVSSv2 temporal score (characteristics of a vulnerability that change over time but not among user environments)
plugin_type	The general type of plugin check (e.g., local, remote)
plugin_version	The version of the Nessus plugin used to perform the check
cm:compliance-info	Detailed information about the compliance check performed.
cm:compliance-result	The result of the compliance check (e.g., PASSED, FAILED).
cm:compliance-actual-value	Relevant output related to the compliance check.
cm:compliance-check-id	The ID for the compliance check performed.
cm:compliance-audit-file	The name of the `.audit` file that invoked the compliance check.
cm:compliance-check-name	The numeric ID and short name of the compliance check.

The information located in the "**plugin_output**" section contains the actual text output of the Nessus scanner. This data section can be very large for some plugins. Some plugins, such as the software enumerations ones, can return data in excess of 20K bytes. If your XML library does not provide the length of a returned data element, or you are manually developing a parser for the `.nessus` format, be very mindful to test the lengths of your strings and make sure your string manipulation functions handle such lengths.

Many plugins will return a multi-line set of information. In this case, storing the information should convert any "newlines" or "return" characters into a "\n" or "\r" set of characters. By "\n", we mean that if the hex character for newline (0x0a) is encountered, it should be replaced by a "\" character followed by "n".

About Tenable Network Security

Tenable Network Security is relied upon by more than 20,000 organizations, including the entire U.S. Department of Defense and many of the world's largest companies and governments, to stay ahead of emerging vulnerabilities, threats and compliance-related risks. Its Nessus and SecurityCenter solutions continue to set the standard to identify vulnerabilities, prevent attacks and comply with a multitude of regulatory requirements. For more information, please visit www.tenable.com.

GLOBAL HEADQUARTERS

Tenable Network Security
7021 Columbia Gateway Drive
Suite 500
Columbia, MD 21046
410.872.0555
www.tenable.com