

NessusClient 3.2 File Format

**March 12, 2008
(Revision 3)**

The newest version of this document is available at the following URL:
http://www.nessus.org/documentation/dot_nessus_file_format.pdf

Table of Contents

TABLE OF CONTENTS	2
INTRODUCTION	3
OVERVIEW	3
FILE STRUCTURE	3
TARGETS SECTION	4
POPULATED TARGETS EXAMPLE	5
POLICIES SECTION	5
PREFERENCES COMPONENT	6
<i>ServerPreferences Element</i>	7
<i>PluginsPreferences Element</i>	7
PLUGINSELECTION COMPONENT	8
<i>FamilySelection Element</i>	8
<i>IndividualPluginSelection Element</i>	9
POPULATED POLICIES EXAMPLE	10
REPORT SECTION	10
REPORTHOST COMPONENT	11
<i>ReportItem Element</i>	12
POPULATED REPORT EXAMPLE	13
ABOUT TENABLE NETWORK SECURITY	15

Introduction

Welcome

Welcome to Tenable Network Security's **NessusClient 3.2 File Format** document. Please share your comments and suggestions with us by emailing them to support@tenablesecurity.com.

This document covers the file format structure for the *.nessus* file, which was introduced with Nessus 3.2 and NessusClient 3.2.

Additionally, an XSD file is available at <http://www.nessus.org/documentation> as well as at <http://blog.tenablesecurity.com/2008/01/introduction-to.html>.

A basic understanding of the Nessus Vulnerability scanner is assumed.

Standards and Conventions

Throughout the documentation, filenames, daemons, and executables are indicated with an italicized font such as *gunzip*, *httpd*, and */etc/passwd*. For emphasis, some words will be **bold faced** to draw attention to them.

Command line options and keywords are printed with the following 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/sc3/daemons
```



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. 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.

File Structure

The *.nessus* file format lists three sections named "Targets", "Policies", and "Report". Each section can have multiple components. A basic outline is shown below, including the "NessusClientData" header and footer:

```
<NessusClientData>
  <Targets>
    # The list of the targets is here
  </Targets>

  <Policies>
    # The list of policies is here
  </Policies>

  <Report>
    # The reports go here
  </Report>
</NessusClientData>
```

It is important to realize that a single *.nessus* file might only contain a targets policy, a scan policy, a scan policy with reported results, or have all sections populated.

Targets Section

The "Target" component with the "Targets" section is made up of zero or more of the following types:

"hostname" type of target:

```
<Target>
  <selected>[yes|no]</selected>
  <type>hostname</type>
  <value>[target]</value>
</Target>
```

"range" type of target:

```
<Target>
  <selected>[yes|no]</selected>
  <type>range</type>
  <start>[target_start_range]</start>
  <end>[target_end_range]</end>
</Target>
```

"network" type of target:

```
<Target>
  <selected>[yes|no]</selected>
  <type>network</type>
  <network>[target_network]</network>
  <netmask>[target_end_netmask]</netmask>
```

```
</Target>
```

For example, the following tells the client to display the target 127.0.0.1 but to UNCHECK the checkbox next to its name:

```
<Target>
  <selected>no</selected>
  <type>hostname</type>
  <value>127.0.0.1</value>
</Target>
```

Populated Targets Example

Here is an example *.nessus* file which specifies a target host name of *www.mycompany.com*, the network range from 192.168.0.0 through 192.168.255.255 and the network of 10.0.0.0/8. Note that the "Policies" and "Report" section have been omitted for brevity.

```
<NessusClientData>
  <Targets>
    <Target>
      <selected>no</selected>
      <type>hostname</type>
      <value>www.mycompany.com</value>
    </Target>
    <Target>
      <selected>no</selected>
      <type>range</type>
      <start>192.168.0.0</start>
      <end>192.168.255.255</end>
    </Target>
    <Target>
      <selected>no</selected>
      <type>network</type>
      <network>10.0.0.0</end>
      <netmask>255.0.0.0</netmask>
    </Target>
  </Targets>
  # Policies and Report sections removed for brevity
</NessusClientData>
```

Policies Section

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

```
<Policy>
  <policyName>[name]</policyName>
```

```

<policyComments>[comments]</policyComments>
<uuid>[UUID]</uuid>
<Preferences>
  <ServerPreferences>
    # Server preferences removed for brevity
  </ServerPreferences>
  <PluginsPreferences>
    # Plugin preferences removed for brevity
  </PluginsPreferences>
</Preferences>
<PluginSelection>
  <FamilySelection>
  </FamilySelection>
  <IndividualPluginSelection>
  </IndividualPluginSelection>
</PluginSelection>
</Policy>

```

- **policyName** is the name of the policy.
- **policyComments** is the comments associated to this policy.
- **uuid** is a UUID attached to this policy. It is used when doing a cross reference with the scan results. Note that this item does not exist in version 3.0 of the Mac OS X client.

Password Attributes

The “Policy” component may have the “**passwordsType**” attribute, which specifies the encoding of the passwords. This attribute can have the following values:

- **Clear Text:** The passwords are stored in clear text.
- **Mac OS X:** The passwords are stored in the Mac OS X keychain. The values in the policy are the UUID of the passwords.
- **Linux:** The passwords are stored in the user home folder. The values in the policy are the UUID of the passwords.
- **Windows:** The passwords are ciphered with the appropriate Windows function. The value in the policy are the ciphered passwords themselves.

For example, if NessusClient has enabled saving of passwords in clear text, the “Policy” will have the attribute “passwordsType” set as shown below:

```

<Policy passwordsType="Clear Text">
  ...
</Policy>

```

If no encoding is specified, the client will try to use its default secure encoding with the passwords.

Preferences Component

The “**Preferences**” component within the “Policy” component contains two elements: “**ServerPreferences**” and “**PluginPreferences**”.

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 “PluginsPreference” element does not need any “item” sections.

Below is an example template for an “item”:

```
<item>
  <fullName>[PreferenceNameAsSentByNessusd</fullName>
  <preferenceName>[Parsed Name]</preferenceName>
  <preferenceType>[entry|radio|checkbox|file|password]</preferenceType>
  <pluginName>[theNameThePreferenceIsAttachedTo]</pluginName>
  <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. Also keep in mind that if the “**preferenceType**” variable is set to “**password**”, then it is **not** saved on disk (it would be considered as 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>
    <fullName>Ping the remote host[entry]:TCP ping destination
port(s) :</fullName>
    <preferenceName>TCP ping destination port(s) :</preferenceName>
    <pluginName>Ping the remote host</pluginName>
    <preferenceType>entry</preferenceType>
    <preferenceValues>built-in</preferenceValues>
    <selectedValue>built-in</selectedValue>
  </item>
  <item>
    <fullName>Ping the remote host[checkbox]:Do an ARP ping</fullName>
    <preferenceName>Do an ARP ping</preferenceName>
    <pluginName>Ping the remote host</pluginName>
    <preferenceType>checkbox</preferenceType>
    <preferenceValues>yes</preferenceValues>
    <selectedValue>yes</selectedValue>
  </item>
  <item>
    <fullName>Ping the remote host[checkbox]:Do a TCP ping</fullName>
    <preferenceName>Do a TCP ping</preferenceName>
    <pluginName>Ping the remote host</pluginName>
    <preferenceType>checkbox</preferenceType>
    <preferenceValues>yes</preferenceValues>
    <selectedValue>yes</selectedValue>
  </item>
  <item>
    <fullName>Services[radio]:Test SSL based services</fullName>
    <preferenceName>Test SSL based services</preferenceName>
    <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 “**IndividualPluginSelection**”. 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 minimize the size of your *.nessus* file considerably.

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 which 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 “Policies” section:

```
<Policies>
  <Policy>
    <policyName>My Example Policy</policyName>
    <policyComment>This is an example policy</policyComment>
    <uuid>CEBCD9B7-5E03-4415-8D37-A90A44A34B9D</uuid>
    <Preferences>
      <ServerPreferences>
        <preference>
          <name>max_hosts</name>
          <value>30</value>
        </preference>
      </ServerPreferences>
      <PluginsPreferences>
        <item>
          <fullName>Services[entry]:Wrapped service read timeout :</fullName>
          <preferenceName>Wrapped service read timeout :</preferenceName>
          <pluginName>Services</pluginName>
          <preferenceType>entry</preferenceType>
          <preferenceValues>2</preferenceValues>
          <selectedValue>2</selectedValue>
        </item>
      </PluginsPreferences>
    </Preferences>
    <PluginSelection>
      <FamilySelection>
        <FamilyItem>
          <FamilyName>Web Servers</FamilyName>
          <Status>disabled</Status>
        </FamilyItem>
      </FamilySelection>
      <IndividualPluginSelection>
        <PluginItem>
          <PluginId>10796</PluginId>
          <PluginName>scan for LaBrea tarpitted hosts</PluginName>
          <Family>Port scanners</Family>
          <Status>disabled</Status>
        </PluginItem>
      </IndividualPluginSelection>
    </PluginSelection>
  </Policy>
</Policies>
```

Report Section

The “**Report**” section can contain zero or more report results. It is organized by specific report name and includes a copy of the scan “Policy” and “Targets” section from when the scan was actually performed. This allows individual reports to have a complete record of their scan configurations even if the scan or targets policy is modified.

Below is a template of how the “**Report**” section is formatted:

```
<Report>
  <ReportName>[reportName]</ReportName>
  <StartTime>[startTime]</StartTime>
  <StopTime>[stopTime]</StopTime>
  <Policy>[Policy XML Object]</Policy>
  <Targets>[Targets XML Object]</Targets>
  <PluginSelection>[pluginSelection]</PluginSelection>
  <ReportHost>
    # Removed for brevity
  </ReportHost>
  <ReportHost>
    # Removed for brevity
  </ReportHost>
</Report>
```

The “**Policy**” and “**Targets**” sections are exact copies of the XML text for those policies and list of targets from when the scan was actually performed. This information can be used to facilitate the import of files into a system such as Security Center. The “**PluginSelection**” component is a list of enabled plugins separated by semicolons such as “10335;10443;12345”.

ReportHost Component

The “**ReportHost**” component within the “**Report**” section contains all of the findings for each host including some meta data such as detected OS as well as a summary of vulnerabilities found by severity. It concludes with a list of items (the actual vulnerabilities) named “**ReportItem**” which will be discussed next.

Below is the template for the “**ReportHost**” component:

```
<ReportHost>
  <HostName>[host name]</HostName>
  <startTime>[timestamp]</startTime>
  <endTime>[timestamp]</endTime>
  <netbios_name>[netbios_name | (unknown)]</netbios_name>
  <mac_addr>[mac_addr | (unknown)]</mac_addr>
  <dns_name>[dns_name | (unknown)]</dns_name>
  <os_name>[os_name | (unknown)]</os_name>
  <num_ports>[number of open ports]</num_ports>
  <num_lo>[number of notes]</num_lo>
  <num_med>[number of warnings]</num_med>
  <num_hi>[number of holes]</num_hi>
  <ReportItem>
    # Removed for brevity
  </ReportItem>
  <ReportItem>
    # Removed for brevity
  </ReportItem>
</ReportHost>
```

Note that the “**netbios_name**”, “**mac_addr**”, “**dns_name**”, “**os_name**”, “**num_ports**”, “**num_lo**”, “**num_med**”, and “**num_hi**” sections are optional. 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.

Version 3.2 of NessusClient populates this data as best it can when creating a report. If data is not available from the scan to fill these values, it uses an “(unknown)” value such as:

```
<netbios_name>(unknown)</netbios_name>
<mac_addr>(unknown)</mac_addr>
<dns_name>(unknown)</dns_name>
<os_name>(unknown)</os_name>
<num_ports>7</num_ports>
<num_lo>2</num_lo>
<num_med>0</num_med>
<num_hi>0</num_hi>
```

ReportItem Element

The “**ReportItem**” element is one finding on a given port on a given host. Its structure is as follows:

```
<ReportItem>
  <port>[PortName]</port>
  <severity>[0|1|2|3]</severity>
  <pluginID>[id_of_the_plugins_doing_the_alert]</pluginID>
  <pluginName>[name_of_the_plugin]</pluginName>
  <data>[PluginData]</data>
</ReportItem>
```

The “**port**” value is expressed as the service name of the port in question, and then the actual port number and protocol separated by a slash in parentheses such as “smtp (25/tcp)”. If a well known name of a port is not available, it will be named unknown such as “unknown (19494/udp)”.

The 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.

Having the “**pluginName**” at hand enables the report to be displayed in an elegant way, rather than forcing multiple cross references all over the place.

The information located in the “**data**” 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 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 “\n” character followed by “n”.

Also note that for an open port (plugin #0) the “**data**” section will contain the word “PORT” such as:

```
<data>PORT</data>
```

Below is an example “ReportItem” element:

```
<ReportItem>
  <port>smtp (25/tcp)</port>
  <severity>1</severity>
  <pluginID>10263</pluginID>
  <pluginName>SMTP Server Detection</pluginName>
  <data>\nSynopsis : \n\nAn SMTP server is listening on the remote
port.\n\nDescription : \n\nThe remote host is running a mail (SMTP) server
on this port.\n\nSince SMTP servers are the targets of spammers, it is
recommended you \ndisable it if you do not use it.\n\nSolution :
\n\nDisable this service if you do not use it, or filter incoming traffic
\n\nRisk factor : \n\nNone\n\nPlugin output : \n\n
Remote SMTP server banner : \n220 mystreth.us.nessus.org ESMTP
Postfix\r\n</data>
</ReportItem>
```

Note how carriage returns are translated to “\n” in the report.

Populated Report Example

```
<Report>
  <ReportName>07/02/22 03:56:52 PM - MyPolicy</ReportName>
  <StartTime>Thu Feb 22 15:56:53 2007</StartTime>
  <StopTime>Thu Feb 22 15:57:42 2007</StopTime>
  <PolicyUUID>CEBCD9B7-5E03-4415-8D37-A90A44A34B9D</PolicyUUID>
  <ReportHost>
    <HostName>localhost</HostName>
    <startTime>Thu Feb 22 15:56:53 2007</startTime>
    <stopTime>Thu Feb 22 15:57:42 2007</stopTime>
    <netbios_name>(unknown)</netbios_name>
    <mac_addr>(unknown)</mac_addr>
    <dns_name>localhost.</dns_name>
    <os_name>Mac OS X 10.4.8 (intel)</os_name>
    <num_ports>5</num_ports>
    <num_lo>18</num_lo>
    <num_med>0</num_med>
    <num_hi>1</num_hi>
    <ReportItem>
      <port>smtp (25/tcp)</port>
      <severity>0</severity>
      <pluginID>0</pluginID>
      <pluginName></pluginName>
    </ReportItem>
```

```
<ReportItem>
  <port>nessus (1241/tcp)</port>
  <severity>1</severity>
  <pluginID>10330</pluginID>
  <pluginName>Services</pluginName>
  <data>A TLSv1 server answered on this port\n\n</data>
</ReportItem>
</ReportHost>
</Report>
```

About Tenable Network Security

Tenable, headquartered in Columbia, Md., USA, is the world leader in Unified Security Monitoring. Tenable provides agent-less solutions for continuous monitoring of vulnerabilities, configurations, data leakage, log analysis, and compromise detection. For more information, please visit us at <http://www.tenablesecurity.com>.

TENABLE Network Security, Inc.
7063 Columbia Gateway Drive
Suite 100
Columbia, MD 21046
TEL: 1-877-448-0489
<http://www.tenablesecurity.com>